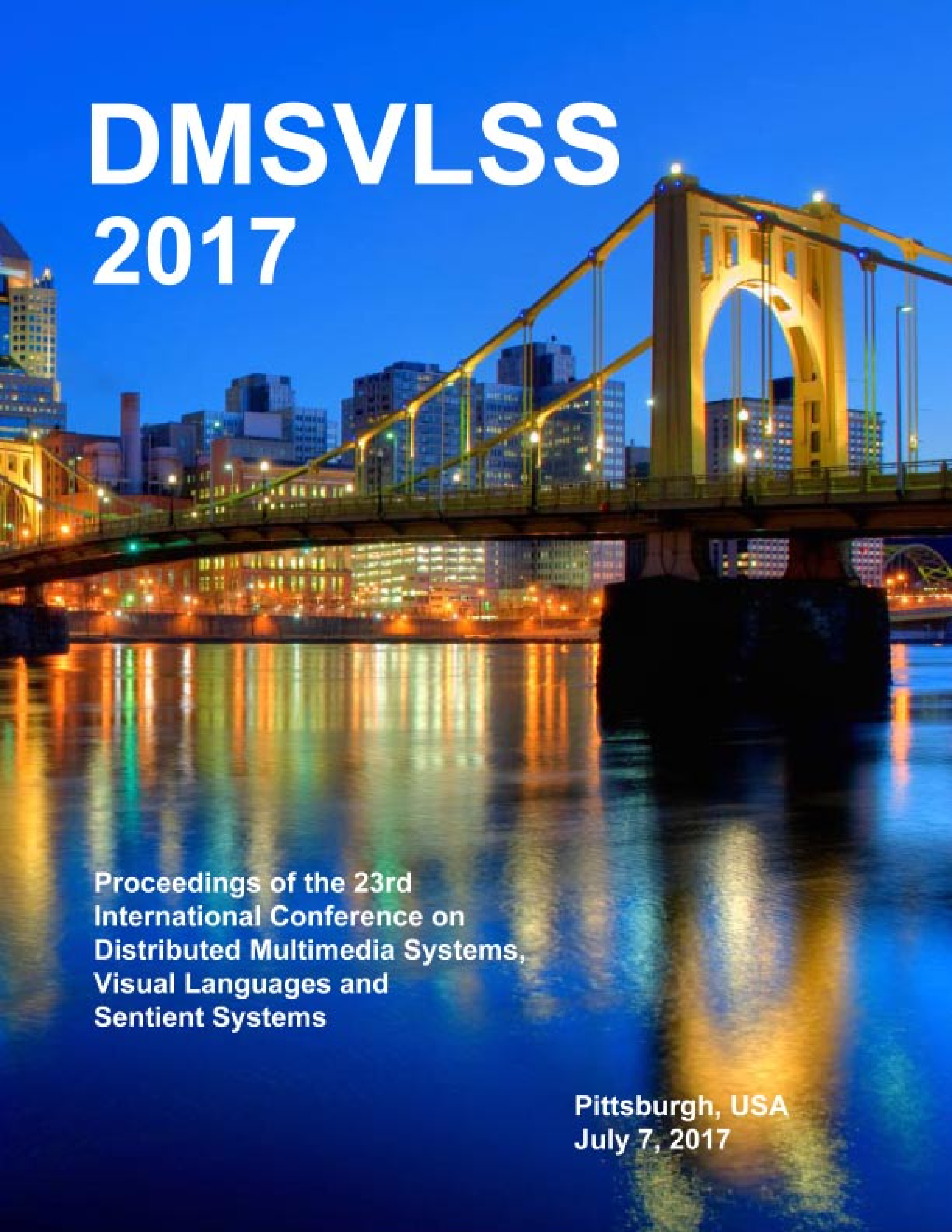


DMSVLSS 2017

Proceedings of the 23rd International Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems

Pittsburgh, USA
July 7, 2017

PROCEEDINGS

DMSVLSS 2017

The 23rd International Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems

Sponsored by

KSI Research Inc. and Knowledge Systems Institute, USA

Technical Program

July 7, 2017

Wyndham Pittsburgh University Center, Pittsburgh, USA

Organized by

KSI Research Inc. and Knowledge Systems Institute, USA

ISBN: 1-891706-42-X

ISSN: 2326-3261 (print)

2326-3318 (online)

DOI: 10.18293/DMSVLSS2017

Additional copies can be ordered from:

KSI Research Inc.

156 Park Square Lane

Pittsburgh, PA 15238 USA

Tel: +1-412-606-5022

Fax: +1-847-679-3166

Email: dms@ksiresearch.org

Web: http://www.ksi.edu/ seke/dmsvlss17.html

Proceedings preparation, editing and printing are sponsored by KSI Research Inc. and Knowledge Systems Institute, USA.

Printed by KSI Research Inc. and Knowledge Systems Institute, USA.

FOREWORD

Welcome to the 23rd International Conference on Distributed Multimedia Systems, Visual Languages, and Sentient Systems (DMSVLSS 2017), which takes place this year in Pittsburgh, Pennsylvania. In this forum, researchers from academia and industry from around the world meet to discuss ideas that involve gathering, processing, interpreting, visualizing, storing, and retrieving multimedia data originating from sensors, robots, actuators, websites, and other information sources.

We are pleased to open the conference with a plenary talk by Giuseppe Polese on Relaxed Functional Dependencies for Big and Multimedia Data Challenges. We also begin this year's conference by recognizing the 2017 JVLC S.K. Chang Best Paper Award winners, Neven A.M. El Sayed, Bruce H. Thomas, Kim Marriott, Julia Piantadosi and Ross T. Smith, for their paper, *Situated Analytics: Demonstrating Immersive Analytical Tools with Augmented Reality* (*Journal of Visual Languages and Computing* 36, 2016, 13-23). The remainder of the conference is organized into four sessions, each of which includes specialized topics: (i) Visual Software Support Tools, (ii) Visual-Aided Data Network Analysis, (iii) Visualization and Computing of Multimedia Data, and (iv) Computational Intelligence, Models, and Algorithms. The conference has a long history of providing a venue for thought-provoking discussions, stimulation of research ideas, and both the initiation of new and the strengthening of existing collaborations. That tradition undoubtedly will continue given the variety of topics that are addressed in this year's collection of papers.

As in the past, paper selection for this year's conference was based upon a rigorous review process. The acceptance rate for full papers was 33%. The conference program contains contributions of high quality research papers and short papers that discuss ongoing research activities and applications from an international population of researchers.

Starting from 2015 The DMS proceedings also contain the Journal of Visual Languages and Sentient Systems (JVLSS). Volume I of JVLSS was published together with DMS2015 Proceedings, and Volume II of JVLSS was published together with DMS2016 Proceedings. We are pleased to announce that starting from 2017 JVLSS will be published independently in a separate volume. Up to 8 papers will be invited and further reviewed for possible inclusion in a JVLSS special issue and/or a JVLC special issue to be published in 2018. Invitations will be made after the DMSVLSS2017 conference.

As Program Chair and Co-Chairs, we would like to express our gratitude and appreciation to the Steering Committee Chair, Dr. S.K. Chang, for his continued support and dedication to the conference, and his invaluable experience. The high quality of the DMSVLSS 2017 technical program would not have been possible without the tireless efforts of many other individuals as well. We would like to thank the Steering Committee for their continuous support and guidance. We also thank the Program Committee whose invaluable, attentive, and timely work has made possible the creation of a high quality technical program. Additionally, we would like to extend our sincere appreciation to all of the authors who submitted their papers to the conference, thereby contributing, through their work and ideas, to the success of this forum. Last but not least, we would like to acknowledge the important contribution of the KSI Research staff whose assistance and support has been truly remarkable throughout the entire organization process.

On behalf of the Program Committee, we are delighted to extend to you our warm welcome to the 23rd International Conference on Distributed Multimedia Systems, Visual Languages, and Sentient Systems (DMSVLSS 2017). We hope that you will find this year's conference a stimulating environment for exchanging ideas, and an opportunity to network with interesting people. Enjoy your visit to Pittsburg, and the United States!

Jennifer Leopold, DMSVLSS 2017 Program Chair
Francesco Colace, Weibin Liu and Chaman Sabharwal, DMSVLSS 2017 Program Co-Chairs

DMSVLSS 2017

The 23rd International Conference on Distributed Multimedia Systems, Visual Languages and Sentient Systems

July 7, 2017

Wyndham Pittsburgh University Center, Pittsburgh, USA

Conference Organization

DMSVLSS'17 Conference Chair and Co-Chair

Giuseppe Polese, University of Salerno, Italy; Conference Chair
Vincenzo Deufemia, University of Salerno, Italy; Conference Co-Chair

DMS'16 Steering Committee Chair

Shi-Kuo Chang, University of Pittsburgh, USA; Steering Committee Chair

DMSVLSS'17 Steering Committee

Paolo Nesi, University of Florence, Italy; Steering Committee Member
Kia Ng, University of Leeds, UK; Steering Committee Member

DMSVLSS'17 Program Chair and Co-Chairs

Jennifer Leopold, Missouri University of Science & Technology, USA; Program Chair
F. Colace, University of Salerno, Italy; Program Co-Chair
Weibin Liu, Beijing JiaoTung Univ., China; Program Co-Chair
Chaman Sabharwal, Missouri University of Science & Technology, USA; Program Co-Chair

DMSVLSS'17 Program Committee

Flora Amato, Univ. of Salerno, Italy
Arvind K. Bansal, Kent State University, USA
Loredana Caruccio, University of Salerno, Italy
Alfredo Cuzzocrea, ICAR-CNR and University of Calabria, Italy
Andrea De Lucia, Univ. of Salerno, Italy
Tiansi Dong, Bonn-Aachen International Center for Information Technology, Germany
Martin Erwig, Oregon State University, USA
Kaori Fujinami, Tokyo University of Agriculture and Technology, Japan
David Fuschi, Brunel University, UK
Ombretta Gaggi, Univ. of Padova, Italy
Angela Guercio, Kent State University, USA
Carlos A. Iglesias, Intelligent Systems Group, Spain
Yau-Hwang Kuo, National Cheng Kung University, Taiwan
Fuhua Lin, Athabasca University, Canada
Alan Liu, National Chung Cheng Univeristy, Taiwan
Max North, Southern Polytechnic State University, USA
Antonio Piccinno, Univ. of Bari, Italy
Giuseppe Polese, University of Salerno, Italy
Genny Tortora, University of Salerno, Italy
Atsuo Yoshitaka, JAIST, Japan
Ing Tomas Zeman, Czech Technical University, Czech Republic

Subcommittee on Distance Education Technologies

Maiga Chang, Athabasca University, Canada
Mauro Coccoli, University of Genova, Italy
Angelo Gargantini, University of Bergamo, Italy
Angela Guercio, Kent State University, USA
Pedro Isaias, University of Queensland, Australia
Hong Lin, University of Houston-Downtown, USA
Paolo Maresca, University Federico II, Napoli, Italy
Elvinia Riccobene, University of Milano, Italy
Michele Risi, University of Salerno, Italy
Veronica Rossano, University of Bari, Italy

Subcommittee on Visual Languages and Computing

Danilo Avola, University of Rome, Italy
Paolo Bottoni, Universita Sapienza, Italy
Peter Chapman, University of Brighton, UK
Kendra Cooper, University of Texas at Dallas, USA
Gennaro Costagliola, University of Salerno, Italy
Sergiu Dascalu, University of Nevada, USA

Publicity Chair

Plenary Talk

Relaxed Functional Dependencies for Big & Multimedia Data Challenges

Professor Giuseppe Polese
Dipartimento di Informatica
Data Science and Technologies Laboratory
University of Salerno, Italy

Abstract: While multimedia data have always been targeted as 'Big', hence advocating efficient techniques for large-scale data processing, with the advent of social networks and other modern applications the term 'Big' Data. and the many related challenges are becoming a main concern also in the context of Enterprise Information Systems, given the tremendous growth in the Volume of data (4300% estimated from 2009 to 2020), together with their Variety, and generation Velocity. In this scenario, since enterprise applications are evolving from the management of structured alphanumeric data only to a broader variety of unstructured data, especially those exchanged over social networks, past challenges and experiences in the context of multimedia data can contribute to solve several problems of big data management in Enterprise Information Systems. To this end, relaxed data dependencies, defined also to improve the organization of multimedia data, can be exploited to tackle several big data management issues. I will introduce and classify Relaxed Functional Dependencies, describe the available techniques to automatically discover them from data, and show some applications in the big data context, such as data cleansing, query relaxation, and view synchronization upon schema evolutions.

About the Speaker: Giuseppe Polese is Professor of Computer Science at the University of Salerno, Italy, and Director of the Data Science and Technologies Laboratory. His research interests concern the areas of Data Science, Multimedia Databases and Web Engineering, with several interdisciplinary contributions to medical informatics and construction engineering. He has published about 100 papers, some of which in top scientific journals. He is an ACM and IEEE member, and member of the editorial boards of International Journal of Software and Knowledge Engineering (2007-present), area editor for Database and Decision Support Systems, and the Journal of Data Science and Engineering (2016-present). Previously, he was a project manager at the Italian Airspace Company, Alenia, consultant for several software firms, including Siemens, Olivetti, and Telecom Italia. He has directed several publicly funded research projects and projects with industry grants.

Table of Contents

Note:
(S) indicates a short paper.
(P) indicates a poster.

Gitsubmit and VeCVL: Integrating Version Control in Introductory Computer Science Education

Nathan W. Eloe
School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468, USA
nathane@nwmissouri.edu

Abstract

Version control systems (VCS), such as Subversion and Git, are pervasive in industry; they are invaluable tools for collaborative development that allow software engineers to track changes, monitor issues, merge work from multiple people, and manage releases. These tools are most effective when they are a part of a developer's habitual workflow. Unfortunately, the use of these powerful tools is often taught much later in a developer's educational career than other tools like programming languages or databases. Even an experienced student's first experience with version control can be unpleasant. In this work, an assignment submission system built around the Git version control system is introduced and analyzed for usability and suitability for use in entry level computer science classes.

Keywords- *computer science education, education technology, pedagogy, version control, visual language*

1 Introducion

Version control tools and methodologies are essential tools in the increasingly collaborative environment. The size and complexity of many modern software development projects require the talent and time of multiple developers working together. While group projects and collaboration are a mainstay of computer science classes, the tools that are used in industry to promote teamwork are often not introduced until later in a computer science curriculum (such as in a Software Engineering course).

Version Control is most effective when it is interacted with on a regular basis. It should ideally become part of the developer's regular workflow. Introducing such an invaluable tool so late in the curriculum forces students to recreate their workflow by breaking bad habits that have been reinforced through early computer science courses and integrating industry best practices. Additionally, the use of these best practices may or may not be reinforced in future classes, requiring the student to self motivate in maintaining the use of those skills.

This paper examines an implementation of a submission system built around the Git version control system that simplifies the process of interacting with version control to only the important steps in a simple workflow (clone, add, commit, push). The interface developed for students implements the Version Control Visual Language (VeCVL) [4]. Additionally, there are tools for teachers and graders to manage assignments and grade submissions. The process aims to simplify the assignment distribution process and reinforce version control workflow in pedagogy, not just as a topic in a course. The student interface and visual language is analyzed from a usability perspective through an evaluation using the Cognitive Dimensions of Notations [10].

2 Background and Related Work

2.1 Git Workflow

Git [2] is a Distributed Version Control System that exhibits a great amount of flexibility to allow powerful and varied workflows [3, 8] to be designed around it. These workflows primarily differ in their approaches and timings for branching and merging. Beyond the branching, the workflows use the same basic cycle of operations for an individual developer: make changes to the local repository, add the changes to the index, and commit the changes (marking them with a commit message). Once a task has been completed, the developer can sync their changes to a remote server (if using a remote for collaboration or backup). This cycle repeats itself, with the developer pulling changes from a remote repository, making local changes, and pushing the changes to the remote repository.

DOI reference number: 10.18293/DMSVLSS2017-005

2.2 VeCVL and Scaffolding

Gitsubmit was developed in concert with VeCVL [4] as a method of introducing version control in education through *scaffolding* [1], and can be seen as the initial implementation of the visual language, expanded to be a full GUI instead of an icon set. VeCVL is a visual representation of the general steps present in version control system in a way that conveys direction of the changes' movement. Section 3 contains an examination of Gitsubmit and its close ties to VeCVL, as well as a discussion of the deviations from the icon set introduced in [4] to adapt the visual language to a full GUI.

2.3 Git in Education

Version control is a topic that is increasingly being introduced in computer science curricula. Significant amounts of research focus on going beyond introducing version control as a topic in a course to embedding the use of version control into pedagogy [11, 5, 1]. As version control becomes more prevalent in industry, computer science education should move to embrace these technologies and introduce them to students.

Code hosting services such as GitHub are joining in the efforts to educate computer science students about version control by offering programs like GitHub Classroom [6] and offering students free benefits when using their services [7].

2.4 Cognitive Dimensions of Notations

The original 14 Cognitive Dimensions of Notations, as introduced by Green [10] are used to evaluate the usability of an existing interface or appropriateness of the method of information delivery. As the student interface to Gitsubmit is aimed at exposing only the needed functionality for a simple git-based workflow, evaluation of the interface will be done by students using the interface in classes (who may have varying levels of experience with git). Certain dimensions, such as Abstraction Gradient, can only be evaluated by developers proficient with the workflow, who are familiar enough with the steps to know whether more details can be encapsulated.

This paper focuses on the following dimensions:

- Diffuseness/Terseness: how many symbols are needed to express a solution?
- Error-proneness: how well does the interface protect the user from mistakes?
- Hard Mental Operations: How much additional "processing" must a user of the system do to complete a solution? How much additional information is needed that is not tracked by the UI?
- Premature Commitment: Is there a firm ordering of steps needed to express a solution? Can a user go back and correct mistakes?
- Progressive Evaluation: Does the UI provide enough feedback as to the current state of the solution?
- Role-expressiveness: How clear is the meaning of each symbol, and the role it plays in the solution?

Some of the dimensions require analysis from people who are skilled in the domain solution. These include:

- Abstraction Gradient: How much can be abstracted by the notation?
- Closeness of Mapping: How well does the notation correspond to the problem?

3 Gitsubmit

Gitsubmit is a submission system for programming assignments with three main parts: a student interface, an instructor interface, and a Git hosting system. What follows is a brief discussion of the hosting system and instructor interface, and an in-depth analysis of the student UI.

3.1 Hosting System

Currently, Gitsubmit uses a self-hosted instance of Gitlab [9] as the hosting backend. This hosting solution was chosen to allow full control of user creation, authentication, and project visibility. The system itself is not tied to Gitlab, and could be modified easily to use other well known hosting solutions such as GitHub or BitBucket. This would require cooperation from these respective companies to set up this functionality, but is an attractive option, as this would move the host system administration away from the instructor.

3.2 Instructor Interface

The instructor interface is a simple interface to automate the creation of a class, the assignment of a project to a class, and the fetching of the student submissions for a specific assignment. The interface is not designed to be a replacement for a full-fledged git client (and indeed abstracts all of the git operations away from the instructor). When discussing the different functionalities of the instructor UI, Gitlab terminology is used; where possible the corresponding concepts from GitHub and BitBucket have been provided.

3.2.1 Creating a Class

A class in Gitsubmit maps to the concept of a *group* in Gitlab (similar to Organizations in GitHub or Projects in BitBucket). The instructor provides the semester, course number, and course name, and a new group is created with the instructor added as a group administrator. The group name is formatted to contain this information (for example, the group W2017.44242.Data_Structures is the Data Structures class (course number 44-242 at Northwest Missouri State University) that is running during the Spring/Winter 2017 semester). Graders and TAs can be added to this group as group administrators as well (allowing them automatic access to student submissions for grading and assistance).

The instructor also provides a CSV file containing student emails and student names. The UI creates a roster in a git repository that contains a mapping of all students in the class to their Gitlab user. If any student does not exist in Gitlab, a user is automatically created. This is one of the differences between GitHub Classroom and Gitsubmit; Gitsubmit removes the need for the student to create their own user on the system. One of the stated goals of Gitsubmit is to remove the parts of the process that are not directly tied to integrating the workflow into the students' everyday development cycle. It also allows a consistent naming convention for student user IDs.

3.2.2 Creating an Assignment

To create an assignment in the instructor interface, the teacher selects a class, names the assignment, and provides either a skeleton directory or a Markdown formatted project description. Optionally, the instructor may also specify a CSV of groups (based on student ID) if the project being assigned is a group project.

The interface fetches the roster from the Gitlab, and creates a repository in the Gitlab group for each individual or student team. The skeleton or description is then pushed to each repository, and the appropriate students are added to the repositories with the Developer role; this allows the pupil to push and pull code from the repository, but not change access permissions (and allow other students to see their submission or working progress). Finally, the interface creates a single repository that contains every student repo as a git submodule. This repository enables efficient fetching of the student submissions with only a few git commands.

This assignment structure is one way Gitsubmit differentiates itself from GitHub Classroom; the assignment structure is designed to not give students the ability to modify the permissions on the assignments. In this way, assignment confidentiality is preserved.

3.2.3 Fetching Student Submissions

Fetching student submissions can be done easily from a command line with four simple commands:

```
git clone <url_of_grading_repo>
cd <grading_repo>
git submodule update --init --recursive
git submodule foreach \
    git pull origin master
```

In order to remove the need for the instructor to drop to the command line, the instructor interface allows the teacher or grader to select an assignment for a class and download all student repositories for that assignment.

3.3 Student Interface

The student interface (Figure 1) was designed to be a very simplified Git client that supports the basic operations needed to use Git as a submission system: clone, push, pull, add, and commit. The icons and UI were designed in tandem with VeCVL [4].

To begin an assignment, the student selects the semester, course, and the assignment (Figure 2). The list of commits in the repository is shown in the right-most pane. To begin the selection, the student clicks the Clone/Pull button (circled in Figure 2). Colors as well as icons are used to indicate the status of the commits. Commits that reside on the server but not locally are indicated with an orange (or red if the commit is the HEAD of origin/master) ID and the Clone/Pull icon. Commits that reside only on the local machine are blue and indicated with the Push icon. If a commit is indicated with a green ID and a check icon, the commit is the HEAD commit on the remote repository and also exists on the local repository (and is the commit that will be graded). Gray commit IDs are common to both local and remote.

After an assignment is started, the student can complete the work normally; using whatever IDEs or other tools are used in the course. As files are modified or added, they appear in the "Unstaged Changes" pane. Students can select which changes should be submitted. The changes are chosen using the add button (circled in Figure 3). Once the student has selected the changes to submit, a commit message can be specified describing what the changes were, and the commit finalized by hitting the Commit button (as circled in Figure 4).

When the student stops working on a n assignment, they can push their work to the server with the Push button (circled in Figure 5). Note the colors indicating the states of the commits in the repositories. The student finalizes their submission by clicking the push button. A successful submission is indicated with a green check mark next to the

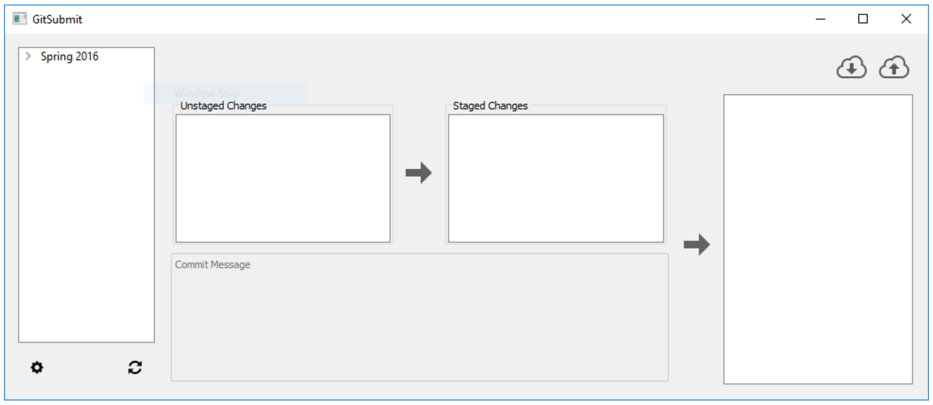


Figure 1. The Gitsubmit Main Window

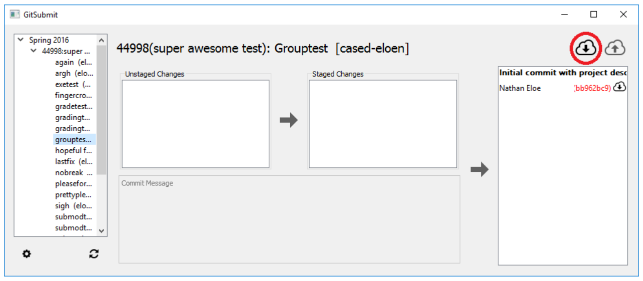

GitSubmit
Spring 2016
44998(super awesome test): Grouptest [cased-eloen]
Unstaged Changes
Staged Changes
Commit Message
Initial commit with project desc
Nathan Eloe

Figure 2. Assignment Selection in Gitsubmit

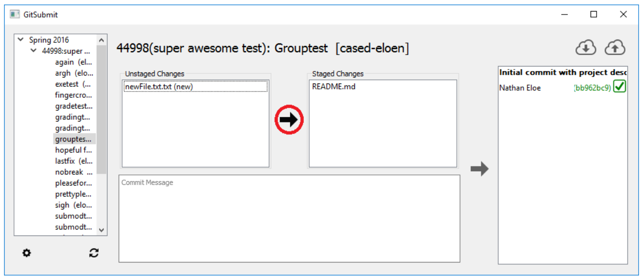


Figure 3. Adding files to the Submission in Gitsubmit

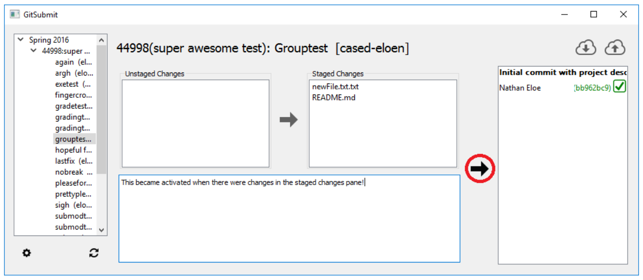

GitSubmit
44998(super awesome test): Grouptest [cased-eloen]
Unstaged Changes
Staged Changes
newFile.txt.txt
README.md
This became activated when there were changes in the staged changes panel

Figure 4. Making a Commit in Gitsubmit

commit the student wants to have submitted (as in Figure 6).

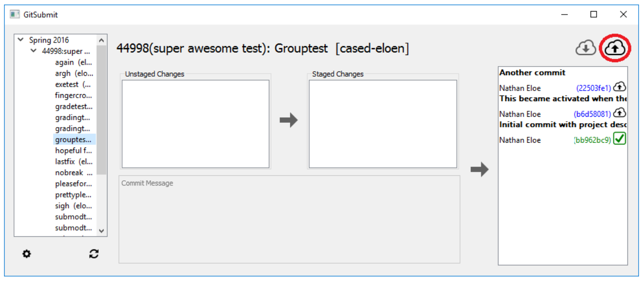

GitSubmit
44998(super awesome test): Grouptest [cased-eloen]
Unstaged Changes
Staged Changes
Commit Message
Another commit
This became activated when the
Initial commit with project desc
Nathan Eloe

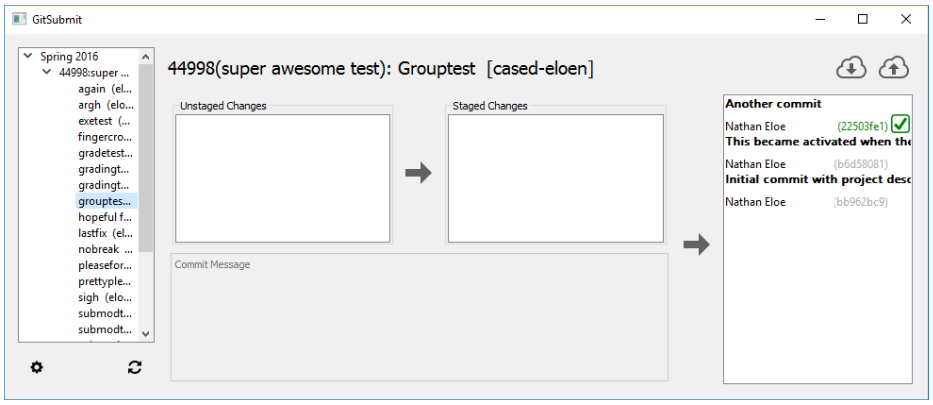
Figure 5. All Commits Ready to Push in Gitsubmit

3.4 Usability Features

Gitsubmit contains some embedded features to help guide the students through the submission process, as well as some visual cues that help form a connection between the submission process and the version control process.

As a student progresses through an assignment submission, options that cannot be performed are disabled; for example, if no files have been added to the staging area or a commit message has not been provided, the button to make a commit is disabled. Actions are only made available to the student when all prerequisites for the step in the submission process have been satisfied. Additionally if there is an indication that the student has not completed the submission process when he or she tries to exit the program (files have not been added to the staging area, or changes have not been committed, or commits have not been pushed), the interface verifies this is intended before closing.

One major stated goal of Gitsubmit is to simplify the introduction to Git and remove steps from the process that are



44998(super awesome test): Grouptest [cased-eloen]

Another commit
Nathan Eloe (22503fe1) ✓
This became activated when the
Nathan Eloe (b6d58081)
Initial commit with project desc
Nathan Eloe (bb962bc9)

Figure 6. Submission Pushed to Gitlab in Gitsubmit

not related to the Version Control workflow. To this end, all interactions that are necessary but auxiliary to the Version Control process are abstracted away. This includes authentication with the central hosting service. When Gitsubmit is first run, the student provides their Gitlab username and password in a first run configuration window (not shown in this paper); this is the only time a student is required to interact with the authentication system. Gitsubmit uses the student provided username and password to obtain an API token from the Gitlab server and generate a SSH key that is used to authenticate and encrypt all Git traffic. This removes the burden of managing authentication methods from the students without requiring that they learn how to generate an SSH key, or provide a password on every push or pull. While authentication is an important part of securing the workflow, it is an external mechanism that is not necessary to understanding the basics of version control.

Tool tips provide both usability enhancement and subtle introduction to the Git verbs. In the assignment select panel, tool tips show additional information (the full name of the project, for example) to help the student determine which project should be chosen. This is particularly helpful when there are multiple similarly named projects that might only differ by the group members (but is not noticeable in the project name itself). The tool tips for the VCS action buttons are the git verbs; in this way, the student begins to make connections between the verbs and the actions those verbs represent in the Version Control process.

4 Usability Evaluation

The survey used to evaluate the usability of Gitsubmit was designed to target the six cognitive dimensions that could be analyzed by a novice in the practice of Version Control. Individuals using the system were asked to evaluate these cognitive dimensions in much the same way one would ask a novice user to perform a Cognitive Walkthrough. Two statements targeted each of the six identified dimensions; one statement approached the dimension from a positive perspective (the UI does X well), while the second looked at it in a negative way (the UI does not do X well). For example, for the dimension of Error-Proneness, the survey gives the following statements:

- The UI makes it easy to make a mistake in the submission process.
- The UI makes it difficult to make a mistake in the submission process.

The exception to this are the statements focusing on Difuseness/Terseness, which asks the user to evaluate two negative statements:

- The UI is not expressive enough to complete a submission.
- The UI is cluttered or complex.

All responses are in the form of a five point Likert Scale, with scores of 1, 2, 3, 4, and 5 corresponding to Strongly Disagree, Disagree, Neutral, Agree, and Strongly Agree, respectively. To determine the UI's overall score for a specific dimension, the scores for the negative and positive statements need to be comparable; as such, the scores for a negative statement are converted to a positive score by determining the distance of the average score from 1 (Strongly Disagree) and taking the score the same distance from 5 (Strongly Agree). This becomes a simple equation:

$$\begin{aligned} adjustedScore &= 5 - (negativeScore - 1) \\ &= 6 - negativeScore \end{aligned}$$

The survey was distributed to sections of classes that have used or are currently using Gitsubmit in their coursework. This includes two sections of a Sophomore level Data Structures class, a Senior level Operating Systems class, and a Junior/Senior level Algorithms class. There was some overlap in students between classes. Of the survey invitations sent out, 34 responses were elicited.

Some of the respondents have experience with Git in other courses (such as a Software Engineering Course) or in industry (through an internship or other professional experience). A portion of the survey asks these students to evaluate Gitsubmit (and VeCVL) on both the Abstract Gradient and the Closeness of Mapping. The statements posed to these more experienced respondents include:

- The UI exposes too many git operations to complete a submission.
- The UI doesn't expose enough git operations to complete a submission.
- The UI abstracts away too many of the git operations.
- The UI should combine more operations into abstractions.
- The UI uses too many symbols to indicate a git operation.
- The UI doesn't use enough symbols to indicate a git operation.

For all of these statements, an average score of less than 3 is a positive indicator.

The full survey can be accessed at `https://www.surveymonkey.com/r/922NT9X`.

5 Results

Figure 7 shows the average score for each statement in the survey aimed at all respondents, as well as the aggregate overall score for each Dimension. For statements posed as a negative, a score below 3.0 indicates that on average students disagree that with the negative statement, and is a desirable score. For positively posed statements, a score above 3.0 shows that Gitsubmit is on average doing well in that category. The reported overall score is an average of the scores of the statements for the given Cognitive Dimension (adjusted in the case of negative statements).

Table 1 shows the average results for the questions directed at students with Git experience. In all cases, the statements were negative, so an average less than 3 reflects well on Gitsubmit and VeCVL. The number of responses (that were not N/A or prefer not to answer) varies from question to question based on student understanding of version control and Git.

Table 1. Average Results for Non-novice Statements. All statements reflected negatively; average scores less than 3 is desirable.

Metric	Avg. Score	Responses
Too Few Operations	2.19	26
Too Many Operations	2.04	26
Too Much Abstraction	2.24	25
Too Little Abstraction	2.71	24
Too Many Symbols	2.08	26
Too Few Symbols	2.31	26

6 Conclusions

The results in Figure 7 show that Gitsubmit as an implementation of VeCVL performs well when analyzed by these eight Cognitive Dimensions of Notations. Overall, students of varying experience levels indicated that Gitsubmit's strongest areas were Diffuseness/Terseness and Progressive Evaluation. This suggests that students find Gitsubmit's interface to be simple enough to use, but provide sufficient functionality to submit the assignment. Students also like that it is able to show them the status of their submission.

The weakest area of those examined is Error-Proneness. Responses were on average slightly positive; this indicates that this is an area where Gitsubmit can improve. Further exploration of the kinds of errors that students are encountering will be needed to determine whether the failings are in the UI, in VeCVL, or both. From an instructor and grader perspective, there have been fewer instances of students submitting the wrong file (or corrupted files) since moving to using Gitsubmit in these classes.

The results in Table 1 are overwhelmingly positive towards both Gitsubmit and VeCVL. The survey indicates that the "Goldilocks Zone" has been reached in terms of number of operations and symbols needed to complete the task. The weakest area for GitSubmit is the amount of abstraction; while students on average agree that there is neither too much or too little abstraction, the results for too little abstraction are closer to neutral than outright disagreement. This indicates an area where Gitsubmit could improve in its usability.

7 Future Work

Gitsubmit and VeCVL are continually evolving works; every course they are used in give feedback and a chance for refinement and improvement. These results show that one area that Gitsubmit could improve in is how well it prevents users from making mistakes. Of particular interest

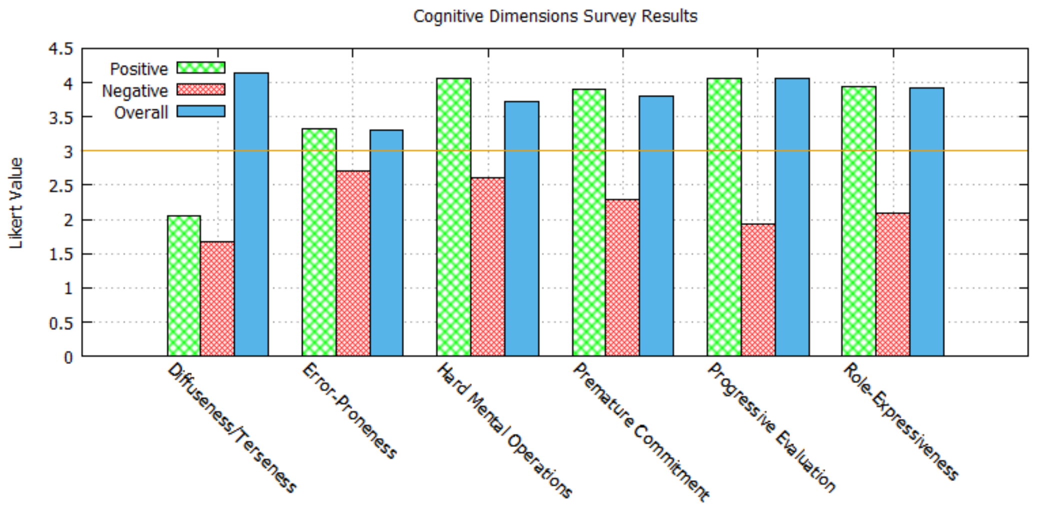


Figure 7. Average Novice Statement Results from Survey. Note that both statements for Diffuseness/Terseness were negatively framed questions; an average score of less than 3 is desirable for both, and both scores were adjusted to determine the Overall score.

is the kind of errors users are making; the notation needs to prevent users from making errors related to submitting the correct files. Most errors that instructors and graders encounter students making relate to the student closing the UI in the middle of a git operation (such as clone or pull), which puts the UI in a state it cannot easily recover from. If students are making other kinds of errors, it needs to be determined whether the notation (VeCVL) or the UI (Gitsubmit) needs to be modified to solve the errors that students are encountering.

Further analysis of the UI is in progress, both through surveys of users and analyses with Human/Computer Interaction tools. A Cognitive Walkthrough analysis is ongoing, and further directed research will investigate the error-proneness of the UI.

Additionally, work is progressing to make the UI look more attractive to users and easier to deploy. Gitsubmit is currently implemented using Python and QT; while development is quick, deployment to multiple platforms (specifically OSX and Windows) is difficult. Updating the UI on student computers is also difficult. Additionally, the UI is not optimal when it comes to interface real estate or visual appeal.

References

[1] D. M. Case, N. W. Eloe, and J. L. Leopold. Scaffolding Version Control into the Computer Science Curriculum. In *Proceedings of the 2016 International Workshop on Distance Education Technology (in conjunction with the 22nd International Conference on Distributed Multimedia Systems (DMS'16))*, 2016.

[2] S. Chacon. *Pro Git*. Apress, Berkely, CA, USA, 2nd edition, 2014.

[3] V. Driessen. A successful Git branching model, 5 Jan. 2010. http://nvie.com/posts/a-successful-git-branching-model.

[4] N. W. Eloe, D. M. Case, and J. L. Leopold. VeCVL: A Visual Language for Version Control. In *Proceedings of the 2016 International Workshop on Visual Languages and Computing (in conjunction with the 22nd International Conference on Distributed Multimedia Systems (DMS'16))*, 2016.

[5] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora. On the Experience of Using Git-hub in the Context of an Academic Course for the Development of Apps for Smart Devices. In *Proceedings of the 21st International Conference on Distributed Multimedia Systems (DMS'15)*, pages 292–299, 2015.

[6] GitHub. GitHub Classroom. https://classroom.github.com.

[7] GitHub. GitHub Education. https://education.github.com/.

[8] GitHub. Understanding the GitHub Flow, 12 Dec. 2013. https://guides.github.com/introduction/flow/.

[9] GitLab. Code, test, and deploy together with GitLab open source git repo management software. https://about.gitlab.com.

[10] T. R. Green. Cognitive dimensions of notations. *People and Computers V*, pages 443–460, 1989.

[11] J. Lawrance, S. Jung, and C. Wiseman. Git on the cloud in the classroom. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 639–644. ACM, 2013.

Package Dependency Visualization: Exploration and Rule Generation

Mubarek Mohammed
Department of Computer Science and Electrical Engineering, Syracuse University
Syracuse, USA
mmohamme@syr.edu

James W. Fawcett
Department of Computer Science and Electrical Engineering, Syracuse University
Syracuse, USA
jfawcett@twcny.rr.com

Abstract— **Large software systems are difficult to understand and complex to manage. Dependency among software components such as packages is one of the reasons that makes software complex. To partly cope with complexity, we designed and implemented a dependency analysis and manipulation tool to manage package dependency at a higher level. Two major graph layout algorithms, spring layout and Sugiyama layout along with clustering algorithms, are used to visualize layout of dependency graphs. In addition, we propose use of layering with Sugiyama algorithm to get insight about software design. It is also possible to generate dependency rules at the architecture level for software designs that are suited for layering design. Both the visualization and rule generation have flexibility to be used with manual restructuring of package dependency.**

Keywords- Visualiztion; Dependency; Layering;[1]

I. Introduction

Many organizations report that they face problems managing software complexity using conventional software engineering techniques [1]. Production sized software may contain millions of lines of code [3]. Understanding, changing and maintaining these large software systems is difficult. Documentation may help reduce complexity to some extent. However, as software evolves, changes may not entirely be captured in documents. Up to date characteristics of the software system may be reflected only in the source code. Extracting information about software from source code is, therefore, important.

Software visualization can be used to manage complexity. Software components can be represented as graphical elements such as node and edges. Graph layout as well as analysis of the graph gives insight about the design of a software system. Specially interesting is the use of node-edge graph to represent dependency of packages.

In this work, we show dependency among components, particularly packages, with node-edge graphs to qualitatively examine overall dependency. Using layered graph drawing, edges are discriminated where those going from upper layers to lower layers are considered generally acceptable edges. Whereas those going in the opposite direction are unwanted edges. We simply use two distinct colors: blue and red respectively.

For cases where distinguishing edges does not give useful information, nodes that result in such edges can be clustered into one of the layers. Strong components algorithm is used to achieve this. Further examination of dependency can be done after clustering to see if the resulting dependency graph makes sense. After examining the dependency, developers may restructure the dependency or leave it as it is. The dependency graph can be further clustered by collapsing nodes in each layer into a single node and adding the corresponding edges among the layers. Rules can be generated, as xml file, from the resulting dependency graph. This may be used to restrict further changes to the existing dependency when change is made to the software system.

The main contributions of this paper are:

- Dependency analysis tool that can be used to get insight about a given software project.
- Layering concept that may potentially be used for planning restructuring package dependency.
- Generation of dependency rules that restricts introduction of unwanted changes.

The remaining part of the paper is organized as follows. Section II introduces software visualization and layering concepts. Sections III briefly discusses the layout algorithms used in our visualization tool. Section IV presents the tool design and brief discussion of each component. Section V shows how the tool can be used in real scenario. Section VI reviews related works in software visualization and dependency. Section VII concludes the paper and future extensions of the work.

II. Software structure visualization and Layering

In this section, we briefly discuss software visualization and layered structure of dependency.

A. Software Structrue Visualization

Software visualization represents software components graphically. Generally, software developers use pictorial representation such as UML diagrams at various stages of software development process. Bringing similar representation at the time of change or maintenance with some interactivity can

[1] 10.18293/DMSVLSS2017-006

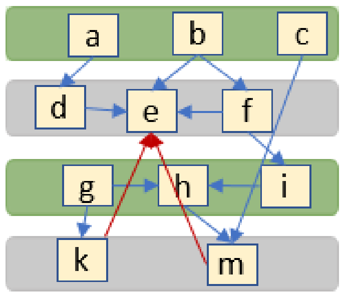


Figure 1. Layering software components.

help understand software better and makes changing or maintenance less complex.

An important concept in software components representation is dependency. A software component is dependent on another component if the former uses instances or services of the later one.

Once dependency analysis is extracted as node-edge graph, it can be drawn after applying layout algorithms. Nodes represent packages, header and/or cpp file in c++. Edges are added if a package depends on another package. Some layout algorithms such as spring layout show visually appealing diagrams that clarify the existing dependency [9]. Other layout algorithms such as the Sugiyama layout algorithm represent such dependency in a hierarchical way [5]. This helps both in understanding and investigating the meaning of hierarchical representation into layers. The next sub-section briefly introduces layering.

B. *Layering*

Layering is representing software components in such a way that lower level layers provide services to upper layers [4]. If software is designed in the form of layers, it can be extended by adding upper layers that use services provided by lower layers and contracted by removal of lower layers.

With a strict hierarchy, each layer uses services in the layer immediately below it. With a non-strict hierarchy, a layer does not have to invoke a service at the layer immediately below it, but it can invoke services at more than one layer below.

Layering of software components results in improved maintainability, reusability or testability. General layering concepts can be shown as in figure 1. There are four layers in the hierarchy, layer 1 at the bottom and layer 4 at the top. Most of the arrows go from upper layers to lower layers. From a to d and c to m are two examples. However, there are arrows going from lower level layers to upper layers indicated by red arrows. Arrows from k to e and m to e are two such examples. As will be discussed later, the later kind of arrows are discouraged in a layered design. Layout algorithms used in this work are discussed in the next section.

III. LAYOUT ALGORITHMS

Layout algorithms arrange nodes in a node-edge graph in such a way that the drawing is visually appealing and readable. Some of the layout algorithms improve symmetry and minimize crossing. Others may arrange the nodes hierarchically.

A. *Sugiyama Layout Algorithm*

Sugiyama layout algorithm is used for drawing flow charts, UML diagrams and other diagrams that needs to be drawn hierarchically [5]. The algorithm has the following steps:

Step 1 - Cycle removal - removes cycles temporarily as the other steps need an acyclic graph. Algorithms starting from the simple depth first search based feedback edge set to the greedy minimum feedback arc set algorithms can be used to make a graph acyclic [6].

Step 2 - Layering - The nodes are assigned layers based on their dependency hierarchy. Independent nodes take the bottom layer and those depending on lower layers take upper layers. Spanning tree algorithm can be used to assign layers. However, restrictions are applied to width and/or height of the layout. A structure like figure 1 can be obtained using layering.

Step 3 – Cross minimization – this step minimizes the number of crossing between edges of different layers. This is done by repeatedly sorting the layered nodes according to barycentric weights calculated from the index position of nodes in the layers. This makes the drawing visually less cluttered.

Step 4 – Coordinate assignment - x and y coordinates are assigned to each node. The y-coordinate of nodes of each layer will be the same. However, the x-coordinate will be assigned based on the index position of a node in each layer.

B. *Spring Layout Algorithm*

Force-directed algorithms model a graph layout problem by assigning attractive and repulsive forces between vertices, and finding the optimal layout by minimizing the energy of the system [7], [8]. The model of Fruchterman and Reigold [9], also known as spring-electrical model, has two forces. The repulsive force, exists between any two vertices, and is inversely proportional to the distance between them. Attractive forces, on the other hand exist only between neighboring vertices (vertices that share an arc) and is proportional to the square of the distance. An example is shown in figure 2. For specific situations spring layout gives the most visually appealing graph. Graph

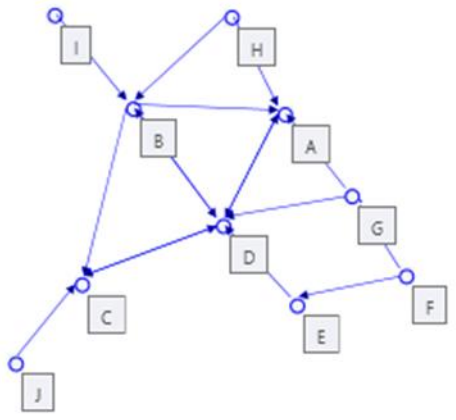


Figure 2. Spring layout example

interaction discussions that follow, except those that require layering concepts, are applicable for spring layout. Therefore, it will not be discussed further.

Before presenting the visualization tool usage, design of the tool developed will be discussed briefly.

IV. System Design and Implementation

To study software dependency understanding and quality related issues, a prototype software visualization and analysis tool has been designed and implemented. The system is depicted in figure 3. The major components are discussed as follows.

Code Analysis - this component analyzes source code and extracts information such as packages, classes, methods and other software constructs that are not used in this work. This part is implemented based on a light weight C/C++ parser implementation [2]. This is a rule based parser which, with minimal change, may be used to parse other programming languages especially those syntactically similar to C++ such as Java and C#.

Graph Processing and Layout - uses information extracted from static analysis as input and results in a node-edge graph. The graph will be further processed to generate the layout. The layout can use spring layout or Sugiyama layout algorithms. In addition, strong components algorithm is used to cluster nodes to modify the layering. Output of this processing is presented as an xml file.

Visualization - this component reads the xml representation of the extracted information with coordinates and renders it on a canvas. The user can interact with the visualization tool to see the type of package represented by the node, panning, dimensional zooming and coordinate based zooming.

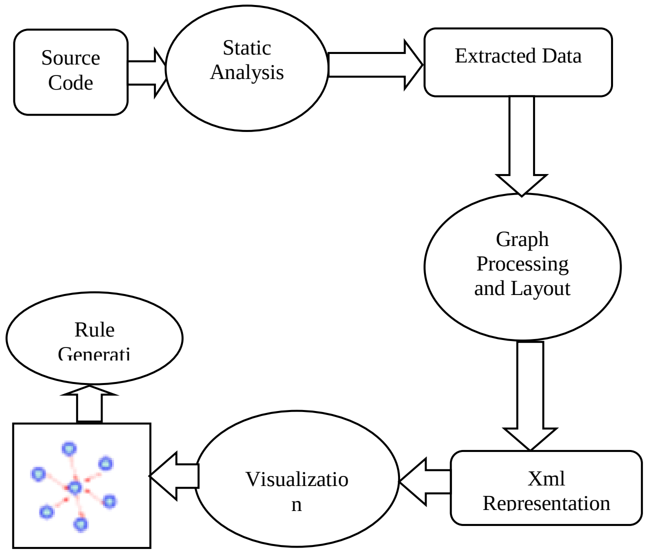


Figure 3. System block diagram.

Rule Generation - unwanted dependencies can be generated as xml rules that may be read automatically, at build time, to prevent further deterioration of dependency. These dependencies can be those not shown by arrows going from upper layers to lower layers. It can also be implicit dependencies that go from lower layers to upper layers. The later ones are red edges if they are drawn on the dependency graph.

The static analysis and graph Processing and Layout components are implemented using C++. Whereas the visualization part is implemented using (Windows Presentation Foundation) WPF.

Even though discussion in the sections that follow focuses on package dependency analysis, the visualization system is general enough to be used for other purposes.

V. Results and Discussions

The dependency analysis tool can be used in different scenarios. Some of its uses are:

- Examining dependency when changing software.
- Getting insight to restructure package dependency.
- Generate rules to prevent addition of unwanted dependency.

We have explored two software systems using the dependency visualization tool:

- Notepad++ - a multipurpose text editor. It has more than 300 packages [10].
- Webkit - taken from chromium web browser source code. It has more than 700 hundred packages [11].

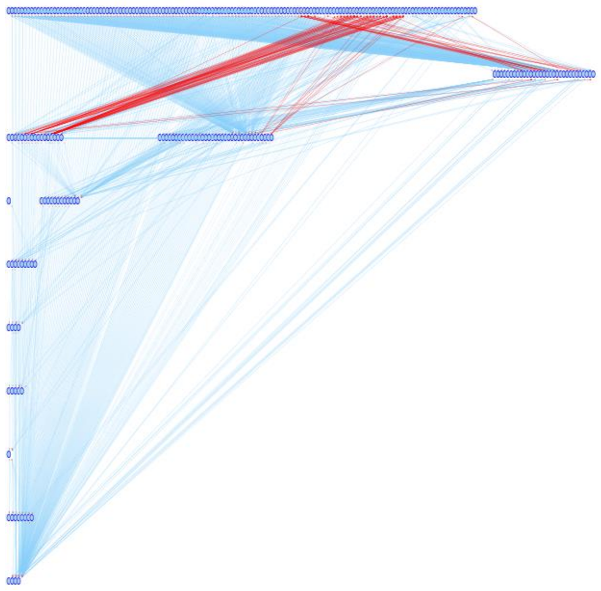


Figure 4. Sugiyama layout of Notepad++ package dependency.

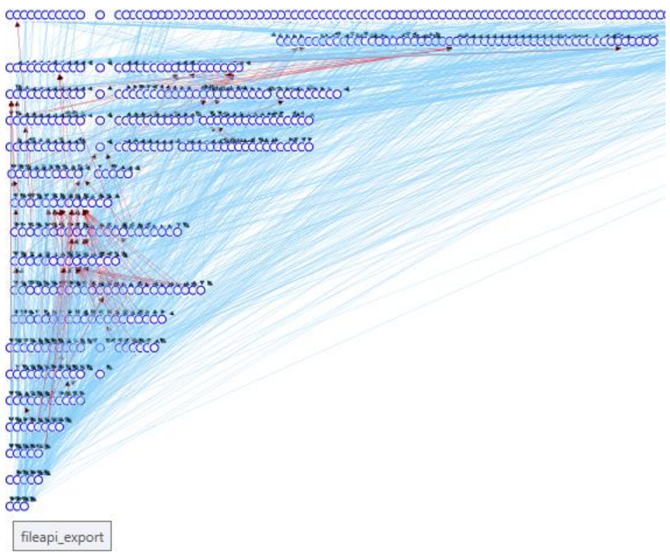


Figure 4. Sugiyama layout of Webkit package dependency.

A. *Exploring Package Dependency Graphs*

Figure 4 and figure 5 show the dependency graph of Notepad++ and webkit respectively. One can explore dependency using features such as zooming, panning and viewing packages names from tooltips. In addition, one can explore areas of interest by selecting and zooming specific selections. In figure 3, the red arrows indicate edges going from lower to upper layers. Whereas blue arrows go from upper to lower layers. Generally, the red arrows are unwanted dependencies in a layered architecture as they result in cycle between layers. Similar observation can be made in Notepad++.

B. *Layout After Applying Strong Components Clustering*

To collapse the red edges into one of the layers, strong components algorithm is run and the layout is redrawn using sugiyama layout. Figure 6 is the resulting dependency graph.

For sake of clarity some parts are clipped. The clustered nodes have more than one component. The tooltip text of one of the nodes is shown as an example. The red edges are significantly reduced. This is due to strong components algorithm clusters nodes in a cycle. Generally, we expect that the red edges to be contained in the clusters.

C. *Clustering Each Layer Into A Node*

Further clustering nodes in each layer results in a graph that shows the relationship between the layers. This is achieved by adding an edge between layers if there is an edge going from any of the nodes from one layer to any other node in another layer. Figure 6 shows what we found for webkit.

Even though layering is generated automatically, after software designers examine its validity and manually restructure the dependency, they can generate rules, as discussed in the next section.

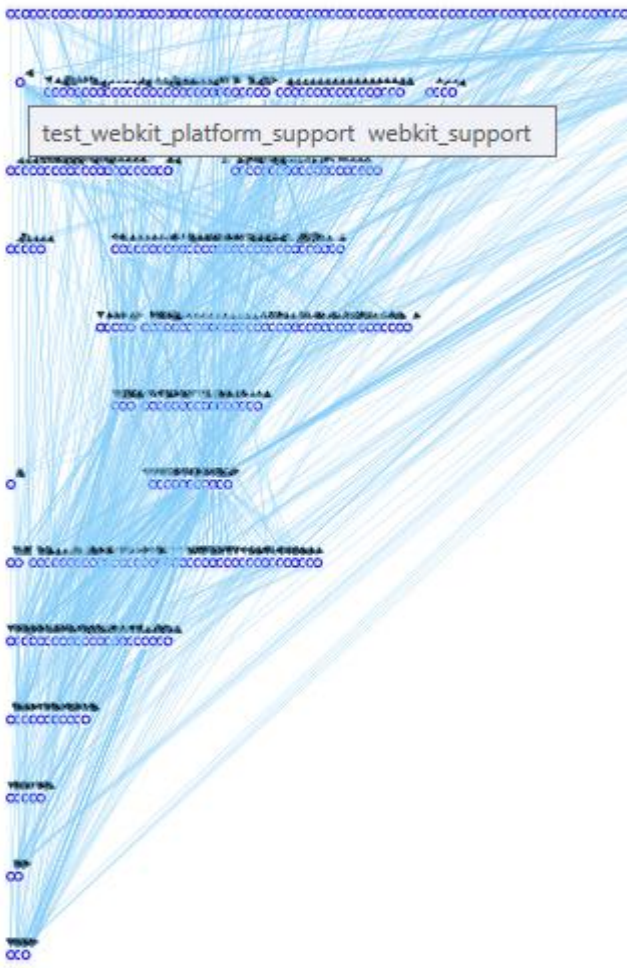


Figure 5. Webkit dependency visualization after applying clustering.

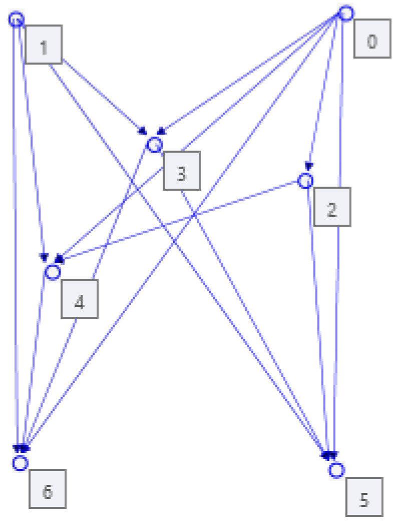


Figure 6. Layer dependency of webkit after clustering packages in each layer.

D. Generating Dependency Rules

If engineers are satisfied with the above layering or the change they made manually after restructuring, they can automatically generate rules similar to figure 8.

Rules can have two parts. The ones shown in figure 8, for example are the ones that do not appear in figure 7. However, we can include implicit rules that prevent addition of dependency from lower layers to upper layers. Note that generating meaningful names for the nodes is beyond the scope of this work

Such rules can be used to notify engineers making change to the software system. This file can be changed by engineers whenever they want to restrict or relax the dependency structure.

```
<? xml version="1.0" encoding="UTF-8"?>
<graph title="Invalid Dependency">
 <node id="0">
    <Edge id="1"/> </node>
 <node id="1">
    <Edge id="2"/> </node>
 <node id="2"> <Edge id="3"/>
   <Edge id="6"/> </node>
 <node id="3">
   <Edge id="4"/> </node>
 <node id="4"> <Edge id="5"/> </node>
 <node id="5">  <Edge id="6"/> </node>
 <node id="6"> </node>
</graph>
```

Figure 7. Dependency rule.

VI. RELATED WORKS

There are many works on software visualization in general. Source code-based visualization is done in [12], [13]. The work in [13] addresses some issues of understanding large industry size software. Class centered visualization is done in [14], [15], [16]. These works represent a class blue print to show the overall structure of a class, control flow among methods, and how methods access attributes. Software organization visualization has been done with optimized visual representation using trees [17], [18]. The organization can also be represented as a treemap [19], [20]. That means containment is defined in rectangular or circular spaces. Another aspect related to organization is concern for software components relationships. The most common ones are Dependency Structure Matrix (DSM), UML class diagrams and Simple Hierarchcal Multi-perspective(SHriMP) [21], [22]. From the three ways, ShHriMP is a relatively complete work. It shows software at source code level, class level and package level.

There are some research works on layering and cycles in software component dependency. Strong components in dependency is studied well in [2]. It emphasizes the fact that strong components make software maintainability and testability difficult. Some research is done to remove cycles to overcome this problem. A heuristic greedy algorithm to find the minimum feedback arc set is used [6]. We used this algorithm at the cycle removal stage of the Sugiyama layering algorithm.

Some research studies hierarchical organization of systems using graph algorithms. Sugiyama layout algorithm is used for hierarchical drawing of graphs [5].

A similar work to ours that visualizes object oriented programs is implemented as a polymeric view in [23]. It is different from our work in that it has a fine-grained view of objected-oriented programs. The approach represents metrics such as weighted calls per method and lines of code graphically. This work is extended in [24] by visualizing software programs at the package level.

In our work, layout algorithms such as spring layout and Sugiyama algorithms are used to show software component dependencies. Interactions such as zooming, panning and saving features allow the user to understand the diagram. Furthermore, Sugiyama layout algorithm is used to hierarchically represent dependencies. The layering and discrimination of edges is used to examine package dependency at a higher level. Strong component clustering along with Sugiyama layout algorithm is used to collapse unnecessary dependencies, red edges that may result in cycle, into a corresponding node in the nearby layer. It is also possible to cluster each layer after which rules to restrict dependency can be generated.

VII. CONCLUSION AND FUTURE WORK

We proposed a dependency visualization tool that can potentially be used to assess software design and generate dependency restriction rules. We implemented two layout algorithms that are used to explore package dependency of real software systems. Features including zooming, panning, pointed and selected zooming, saving layout and printing layouts are some of the functionalities of the tool. In addition, it helps one to explore high level design of a software and possibly help guide the restructuring of package level dependency.

Layering is one of the software architecture focused processes used in software design. The logical layering with additional input from engineers help restructure package dependency. After automatically identifying strong components in package dependency graphs, logically layering packages enables engineers to manage complexity by enabling them to control change in an ordered manner. Clustering packages in a layer after possible manual change of the dependency information results in a layered high level design of the software under investigation. If the layering in the high level layered design is found to be useful, rules can be generated that can be used as configuration file to check software change that prevents deterioration of design. Of course, such rules can be changed to further restrict or relax possible dependencies.

This work is a preliminary result. It has limitations that should be addressed in future work. Some of the planned works are:

- Getting feedback from real users will make the tool useful in real software design restructuring. The future change will be more concrete if we get feedback from engineers using our tool.
- Strong components hide cycles in package dependency. Whenever possible breaking these

cycles improves the design. Allowing automatic suggestions of such restructuring will be very helpful.

- Not all software designs benefit from layering architecture. Allowing other ways of arranging packages to automatically suggest high level design is also useful.

REFERENCES

[1] Ultra –Large-Scale Systems: The Software Challenge of the Future, Retrieved from http://www.sei.cmu.edu/library/assets/uls_Book20062.pdf.

[2] J.W. Fawcett et al., Analyzing static structure of large software systems, proceedings of the 2005 International Conference on Software Engineering Research and Practice, 2005.

[3] T. Ball and S.G. Eick. Software visualization in the large, IEEE Computer, Vol. 29, April 1996, pp. 33–43.

[4] H. Gomma, Software Modeling and Design, Uml, Use cases, Patterns, and Software Architectures, Cambridge Univesity Press, 2011.

[5] K. Sugiyama and et al., Methods for Visual Understanding of Hierarchical System Structures, IEEE Transactions on Systems, Man, and Cybernetics, VOL. SMC- 1, NO. 2, 1981, pp. 109-125.

[6] P. Eades et al., A fast and effective heuristic for feedback arc set problem, Information processing letters, Vol 47, Issue 8, 1993, pp. 319 – 323.

[7] P.Eades. A heuristic for graph drawing. Congressus Nutnerantiunt, 42:149 – 160, 1984.

[8] Hu, Y. F. "Efficient, High-Quality Force-Directed Graph Drawing." The Mathematica Journal 10, no. 1 (2006): 37-71.

[9] T.M.J Fruchterman and E.M. Reigold, Graph drawing by force directed placement, Software – Practice and Experience, 21:1129 -1164, 1991.

[10] Notepad++, retrieved from http://notepad-plus-plus.org/download/v6.2.2.html , sept. 2012.

[11] Webkit from chromium web browser project, retrieved from http://dev.chromium.org/developers/how-tos/get-the-code , sept 2012.

[12] S. Eick, J. Steffen, and E. Summer Jr., “Seesoft – A Tool for Visualizing Line Oriented Software Statistics, “IEEE Trans. Software Eng., vol 18, no. 11, pp. 957-968, Nov. 1992.

[13] T. Ball and S. Eick, “Software Visulization in the Large,” Computer, vol. 29, no. 4, pp. 33-43, Apr. 1996.

[14] M. Lanza, “Object Oriented Reverse Engineering – Coarse-Grained, Fine Grained, and Evolutionary Software Visualization, PhD dissertation, Univ. of Bern, 2003.

[15] M. Lanza and S. Ducasse, “A Catogorization of Classes Based on the Visualization of Their Internal Structure: The Class Blueprint,” Proc. 16th ACM SIGPLAN Conf. Object-Oriented Programming, Systems, Languages, and Applications, pp. 300-311,200.

[16] S. Ducasse and M. Lanza, “The Class Blueprint: Visually Supporting the Understanding of Classes, “IEEE Tans. Software Eng. vol 31, no. 1, pp. 75 – 90, Jan. 2005.

[17] C. Wetherell and A. Shannon, “Tidy Drawings of Trees, “IEEE Trans. Software Eng., vol SE-5, no. 5, pp. 514 -520, Sept. 1979.

[18] T. Barlow and P. Neville, “A Comparison of 2D Visualization of Hierarchies, “Proc. IEEE Symp. Information Visualization, pp. 131-138, 2001.

[19] B. Johnson and B. Shneiderman, “Tree-Maps: A Space-Filling Approach to the Visualization of Hierarchical Information Structures, “Proc. Second IEEE Conf. Visualization, pp. 284 – 291, 1991.

[20] B. Shneiderman, “Tree Visualization with Tree-Maps: 2D Space-Filling Approach, “ACM Trans. Graphics, vol. 11, no. 1, pp. 92 – 99, Jan. 1992.

[21] M. Storey, H. Muller, and W.K., “Manipulating and Documenting Software Structures, “Software Visualization, pp. 244 – 263, World Scientific Publishing Co., 1996.

[22] M. Eiglsperger, “Automatic Layout of UML Class Diagrams: A Topology-Shape-Metrics Approach,” PhD dissertation, Univ. Tubingen, 2003.

[23] R. Francese, M. Risi, G. Scanniello, and G. Tortora, “Proposing and assessing a software visualization approach based on polymetric views,” *Journal of Visual Languages & Computing*, vol. 34–35, pp. 11–24, Jun. 2016.

[24] R. Francese, M. Risi, G. Scanniello, and G. Tortora, “Enhancing Polymetric Views with Coarse-Grained Views,” in 2016 20th International Conference Information Visualisation (IV), 2016, pp. 57–62.

[25] J.W. Fawcett, (2016, December 10), Light Weight Parser [Online]. http://www.ecs.syr.edu/faculty/fawcett/handouts/WebPages/blogParser.htm.

The Design and Evaluation of a Text Editing Technique for Stylus-Based Tablets

Gennaro Costagliola, Mattia De Rosa, Vittorio Fuccella
Dipartimento di Informatica, University of Salerno
Via Giovanni Paolo II, 84084 Fisciano (SA), Italy
{gencos, matderosa, vfuccella}@unisa.it

Abstract

We describe the design and evaluation of a technique aimed at improving text editing on touchscreen devices that exploit the use of stylus-based gestures. The technique has been designed by choosing the most natural gestures for users, established in a preliminary study. The technique allows the user to interact directly with the text to perform commands such as select, move, copy, delete and paste. We conducted an experiment to compare the gestural editing technique to the classical technique (present on Android devices). Results show an advantage in terms of efficiency for the gestural technique with large font.

Keywords: *Gestures; Text editing; Stylus tablet; Touchscreen.*

1 Introduction

Mobile touchscreen devices such as smartphones and tablets are becoming ever more popular in recent years. These devices are used to perform a wide range of operations: web browsing, chatting, reading documents, etc. With the proliferation of touchscreens, the research has focused on improving user interaction with them. At the end of the 60s, Coleman was one of the first authors to study the use of handwritten symbols for text-editing [8]. Other studies dealt with text-processing [17] and sketch-editing [6]. In view of this research, there are increasingly common applications that take advantage of gestures to facilitate some operations. A gesture is a sign made by hand (e.g. a circle, an arrow, etc.) possibly used to denote a command.

A suitable application for gesturing is text editing. While on a classic personal computer mouse/touchpad, keyboard and WIMP-based interfaces are used to perform editing operations with reasonable efficiency, text editing can be particularly difficult on mobile touchscreen devices. In fact, the screen size can create additional obstacles: the use of a finger as a pointing device introduces problems of accuracy and occlusion (only partially solved by using a stylus). Additionally, it is not possible to use keyboard shortcuts (because a physical keyboard is typically absent).

For the reasons mentioned above, text editing is not a very common task on mobile devices. Text editing on a touchscreen is currently performed through a *Widget-based* technique. Typically, the user enters text using a soft keyboard (or handwriting) and moves the cursor by simply tapping with his/her finger on the desired point of the text. Besides entering new text, the user may perform selection and editing operations by using the widgets that appear in a menu over the text after a user interaction (e.g. a long press). In this context, the use of gestures could facilitate this task, for example, a user may delete a word by simply performing a specific gesture (e.g. a cross) over it.

The aim of this paper is to design a new text editing technique based on gestures, which allows performing the main operations in a simple and intuitive way. In order to identify the most natural gestures for text editing, a preliminary exploratory study was carried out. The set of tasks was defined by analyzing some papers on the same topic, including [18, 15, 24]). Starting from the data collected through this preliminary study, we identified the most appropriate gestures for each editing operation. The user can perform them directly on the text, thus making the editing technique independent from the text entry method (e.g. a soft keyboard). This way the user may choose any input method, depending on the context and its preferences, including keyboards that make use of gestures [19, 12, 11, 14].

This paper is organized as follows: Section 2 describes some works related to ours; Section 3 describes the preliminary experiment regarding user preferences on gestural text editing; Section 4 describes the proposed text editing technique and Section 5 its experimental evaluation. A discussion on the limitation of this study and some comments on future work conclude the paper.

DOI reference number: 10.18293/DMSVLSS2017-009

2 Related Work

The introduction of touchscreen has led to the study of new forms of interaction. One of the first metaphors used was to simulate the use of paper, for example to allow drawing of figures or diagrams [1, 16, 9, 10]. Beyond this simple form of interaction, much research in the HCI field has focused on the use of gestures to perform actions. Most of it developed before the rise of finger pointing and multi-touch gestures (e.g. [24, 22]) and was based on the slower but more precise stylus/mouse. Bragdon et al. [4] discovered that in the presence of distractions, the use of gestures to issue commands improves performance w.r.t. the use of touch buttons (such as a QWERTY keyboard). The reason is that using gestures requires less attention than using widgets since in the latter case the user has to look at the keys before touching them. Moreover, when dealing with shortcuts, users tend to remember gestures better than keyboard shortcuts and make fewer mistakes [2].

Findlater et al. [12] pointed out another aspect of the usefulness of gestures. In particular, their study focused on writing non-alphanumeric input using gestures on touchscreen devices equipped with QWERTY soft keyboards. The advantage of this technique is that people do not need to change the interface to enter symbols that are not letters and numbers. The results showed an overall favor for this technique over moded-keyboard interfaces.

Text editing research had its climax at the time of the first graphical user interfaces, starting from the cut/copy-paste technique developed since the 1970s. Text editors have become increasingly complex, leading to studies on the new features [3, 20, 21]. In the literature, there are a few works discussing text editing with gestures on mobile devices. Some works only focus on a single editing function. Chen et al. [7] focus instead on the Copy-Paste operations, that are not as easy to perform on smartphones as on desktop computers. They propose BezelCopy, a copy-paste technique based on bezel-swipe gestures, and evaluate it showing that it outperforms alternative approaches for a number of commonly performed copy-paste tasks. Scheibel et al. [23] focus on the problem of precise pointing on touchscreens, proposing a virtual stick controller technique and evaluate it in a text editing context showing that it may facilitate the placing of the cursor when the font size is small.

Many commercial applications face the same problem with different solutions. In Apple iOS, a magnifying glass appears on the touched text after a long press. Then the user can move his/her finger on the magnified area and the view is updated in real time. This technique avoids the problem of occlusion and allows to correctly position the cursor even if the text has a very small font. Android implements a graphical widget attached to the lower end of the cursor. The widget can be moved with the finger, partially solving the problem of occlusion, but it is not very practical for small fonts and does not offer magnification. Moreover, many soft keyboards offer cursor movement with the help of arrows on the keyboard, e.g. the Hacker's Keyboard[1] and the Arrows Keyboard[2].

Fuccella et al. [15] compare a gesture-based editing technique to a widget-based one. The results show a performance improvement of 13-24% for the gesture technique. The feedback from the participants was also positive. Moreover, the two editing techniques use different input channels, so they can co-exist on a single device. This means that gestural editing can be added to any soft keyboard without interfering with the experience of the user that chooses not to use it.

There are several studies in the literature that evaluate the performance of text editing. The above-mentioned papers include the evaluation of editing techniques on touchscreen devices. Older works, however, have a different focus, such as Wolf and Morrel-Samuels [24] that conducted an experiment on pen and paper for simple text-editing tasks. The purpose of this study was to analyze the consistency and frequency of gestures for editing operations. This experiment was performed with 12 participants and was divided into three phases. In the first phase, participants had to choose a gesture for each task and perform the editing operations. The second phase was repeated immediately after the first one and the participants had to use their chosen gestures again; the last phase, equal to the previous one, was performed after a week. The feedback from participants was positive as shown by answers to a questionnaire: 85% said it had carried out the task of editing without having to think too much about the gesture to use, 69% said that it seemed natural to use gestures and 77% easily remembered previously performed gestures.

3 Preliminary experiment

The aim of this preliminary experiment is to find the most suitable set of gestures for editing text on the basis of users' preferences. This study takes inspiration from the work of Wolf and Morrel-Samuels [24] described in Section 2, with the difference being that in our case the tasks are performed on a touchscreen instead of on a paper.

3.1 Participants

We recruited 10 participants (3 female) between 19 and 25 years old (M=22.9, SD=2.13). All of them were university students who agreed to participate for free. All participants had experience with touchscreen devices and 9 of them also with their use with a stylus.

[1]http://code.google.com/p/hackerskeyboard/

[2]http://arrows-keyboard.android.informer.com/

3.2 Apparatus

The experiment was carried out on a Huawei P8 smartphone running Android 5.0.1. The device has a 5.2” touchscreen which can be operated with both finger and a stylus.

The experimental software was developed in Java in order to allow execution and recording of editing tasks. It consisted of an Android application, showing at the top the description of the requested text editing operation and some text below, with the parts to be edited highlighted in green. The gestures performed by the user are shown over the text as red lines in order to provide a feedback. Since our goal is just to record the gestures, the gestures produce no effect on the text. A *Next* button at the bottom right of the screen allows the user to advance to the next task. For each completed task the software logs the data of the user gestures as a list of (*x, y, time*) tuple and a screenshot of each of them.

3.3 Procedure

Before starting the experiment, participants had an introductory phase where the experimental procedure was briefly explained and some demo tasks were shown. Then they were asked to fill out a pre-experiment questionnaire with the following information: personal data (age, gender); handedness (right-handed, left-handed); previous experience with touchscreen devices (tablets, smartphones, etc.) whether they had experience using a stylus.

After that, each participant was asked to perform the editing tasks proposed by the application, following the shown task descriptions. There was no time limit. We selected thirteen different editing tasks: split word, delete character, select phrase, delete paragraph, delete phrase, delete word, insert character, enter a word, join text, move row, move phrase, move word and select text. Each task was presented twice to each participant, for a total of 26 tasks. The tasks were presented sequentially in a random order.

At the end, a questionnaire was given to each participant to collect information and opinions.

3.4 Design

The experiment was a within-subjects design. The independent variable was the task, with 13 test condition. The dependent variables were 2: the gesture and the task completion time.

The screenshots recorded for each task were analyzed by a human operator in order to classify the editing gesture(s) made by the participant. Since 10 participants performed 13 tasks, each of them twice, $10 \times 13 \times 2 = 260$ screenshots were analyzed to identify the gestures.

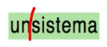
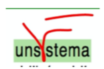
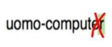
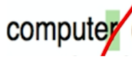
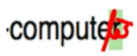
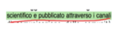
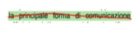
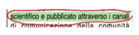
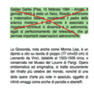
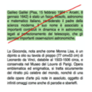
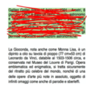
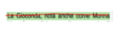
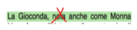
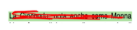
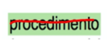
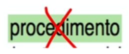
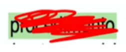
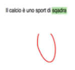
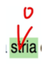
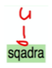
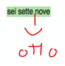
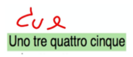
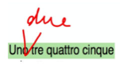
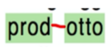
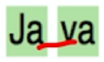
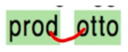
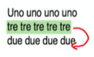
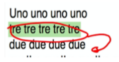
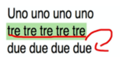
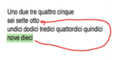
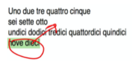
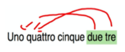
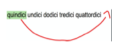
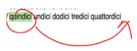
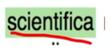
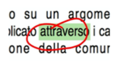
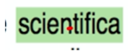
Task	Gestures			Other
split word	 90%	 10%		0%
delete character	 45%	 40%	 10%	5%
select phrase	 75%	 15%	 10%	0%
delete paragraph	 70%	 20%	 10%	0%
delete phrase	 60%	 30%	 10%	0%
delete word	 50%	 30%	 10%	10%
insert character	 65%	 25%	 10%	0%
enter a word	 35%	 30%	 25%	10%
join text	 40%	 25%	 10%	25%
move row	 65%	 25%	 5%	5%
move phrase	 45%	 30%	 25%	0%
move word	 70%	 30%		0%
select text	 55%	 25%	 20%	0%

Table 1: Gestures classes used by the participants to complete each task and their frequency.

3.5 Results and discussion

All participants completed the experiment. Except for introduction and questionnaires, the mean experiment completion times among the participants was 202 seconds (SD=56).

For each task, the gestures used by the participants were identified. Table 1 shows each gesture class frequency (as a percentage) and a gesture example (taken from the experiment screenshots).

It is worth noting that only for a subset of the tasks was it possible to clearly identify a gesture preferred by the participants. In fact, for some tasks, the participants split between different alternatives. An example in which almost all of the participants used the same gesture is the *"split word"* task (9 out of 10 participants performed the same gesture). There was no consensus, instead, for example for the *"delete character"* task.

The final questionnaire indicated that none of the participants spent much effort in thinking of a possible gesture to complete the task, and that in the second trial of a task most of the participants remembered the gesture performed the first time.

4 Design of the Gestural Editing Technique

We designed our text editing technique on the basis of the data collected in the above-described preliminary study. The selection of the set of editing features supported through gestures also took into account the work of Roberts [20]. For most editing actions we selected the first or second most frequent gesture resulting from the previous experiment, taking into account the need to remove the ambiguity between different selected gestures. For the editing actions requiring the entry of some text, we decided not to rely on any specific input method (e.g. handwriting), but to rely on the default system method (usually a soft keyboard). Moreover, in order to support typical text editor operations, gestures for operations like *copy, paste, cut, undo and redo* were added.

The set of supported editing features and related gestures are shown in Table 2. The *Gesture* column shows how the gesture is performed on the text, while the *Explanation* column describes the functionality implemented by that gesture. Every gesture begins with a dot (in red), continues with a line (in black) and ends with an arrowhead (in red). This notation shows all the movements of the stylus from pressure to release, with the arrow indicating the gesture direction. The gesture set can be divided into three categories:

- *deletion*: gestures indicating the deletion of text (individual characters or one or more consecutive words). To delete a character the user can simply draw a diagonal line (slash) over the character to be deleted. To

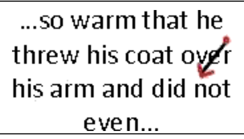
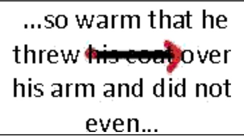
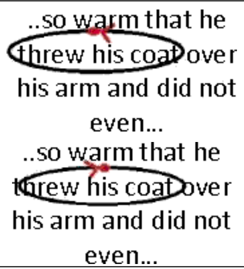
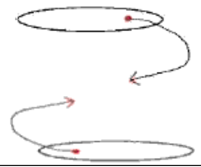
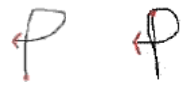
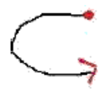
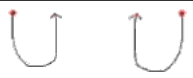
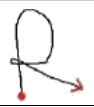
Gesture(s)	Explanation
 ...so warm that he threw his coat over his arm and did not even...	The character underneath the gesture will be deleted.
 ...so warm that he threw his coat over his arm and did not even...	The text underneath the gesture will be *cut* (deleted and copied in the clipboard).
 ..so warm that he threw his coat over his arm and did not even... ..so warm that he threw his coat over his arm and did not even...	The text enclosed within the outlined area will be selected (and highlighted).
	The text selected by enclosing it within the outlined area will be moved to the point where the second gesture ends.
	*Paste* the text from the clipboard (if present) at the place where the cursor is currently placed.
	*Copy* the selected text into the clipboard.
	*Undo* the last performed editing operation (if there is one).
	*Redo* the last operation that has been undone (if there is one).

Table 2: Gesture set implemented by the gestural editing technique. Gesture starts are shown as a red dots, gesture ends as red arrowheads.

delete one or more words, s/he can draw a horizontal line that covers the portion of text to be deleted. If more than half of a word is covered, it will be fully deleted. Word deletion also performs the *cut* functionality, i.e. the deleted text is inserted in the clipboard.

- *moving/copying*: gestures allowing text movements. There are two ways to perform such operations: *cut-paste* and *select-move*. In the former, the user first deletes some text, then places the cursor at the desired point and performs a *paste* function. The deletion is performed as specified at the previous point. The paste operation is performed through a `P` gesture. In the latter, the user first selects some text by drawing an ellipse around it. This highlights the selected text and allows the user to drag it by tracing a line starting from the highlighted area and ending in the place in which the text should be moved. The selected text remains in place and it is moved to its final location only when the gesture ends. Finally, to only copy a piece of text, without cutting or moving it, the user can select it as above and then perform the `C` gesture. To paste the copied text, the same procedure used for the *cut-paste* is used, i.e. placing the cursor and performing the `P` gesture.
- *editing correction*: to correct editing errors, the user can perform the `U` undo gesture. To redo the editing operation that was incorrectly invalidated with an undo, the user may simply perform the `R` gesture.

In order to perform the gesture recognition, the PolyRec [13] gesture recognition method was used.

5 Experiment

To check the effectiveness of the proposed technique, we compared it to one of the standard text editing techniques for mobile devices. In particular, we compared it to the standard text editing technique available on the Android system. During the experiment, we collected information about the user performance in carrying out the proposed editing tasks.

5.1 Participants

For the experiment, 12 participants (5 female) between 21 and 30 years old (M=24.17, SD=2.58) were recruited. All of them were university students who agreed to participate for free, with no overlap with the participants of the experiment described in Section 3.

All of them already had experience with touchscreen devices, using their smartphones every day. They were asked how frequently they perform text editing operations on touch devices, and we found that text editing is rarely performed, except for a single participant who declared to perform it frequently to send emails and messages.

All the participants declared an average knowledge of the English language. We considered this level sufficient to perform editing tasks (of English text).

5.2 Apparatus

The experiment was carried out on a Mediacom tablet running Android 4.4.2. The touchscreen display has a size of 10.1” and a resolution of 1366x768 pixels. A simple capacitive stylus was used to interact with the device.

The experimental software is an Android application showing a list of all the tasks to be performed. By tapping on one item, the corresponding task is launched in an editing view. After the user completes a task, its entry in the list is highlighted. The editing view uses either the traditional editing widget (EditText Android widget) or the gestural editing technique (a custom version of the Android TextView widget). A task is not considered completed if the text still contains errors. Nevertheless, the user had the option to abandon a task (e.g. if the text had been irreparably changed). In this case, the task was reset and restarted.

The software records all major user actions and calculates the completion time of each task. The task is considered completed when the current text equals the text defined in the task solution. At task completion, an *Android toast* with the completion time is shown to the user.

5.3 Procedure

Before starting the experiment the participants were asked to fill out a form with the following information: age, gender, English knowledge level, experience level with touchscreen devices and with text editing on such devices. Then they received an explanation of the experimental procedure and were given an instruction sheet, containing a table with the set of allowed gestures (in gestural editing mode) and a table with the task list (including the initial text and correct text). The sheet was left with the user throughout the whole experiment.

The experiment started after the experimenter ascertained that the participant had well understood the procedure. The experiment consisted of seven tasks in which the user had to correct the given text, each one repeated for the two editing techniques. The experiment was divided into four blocks. The first block was used as user training (not recorded) and performed with the medium font size. Half the users first used the gestural editing technique and then the classic technique, while the other half followed the reverse order. The other three blocks composed the actual experiment, each one with a different font size (small, medium, large).

Task	Title	Description	Presented Text	Final Text
1	Delete character	Delete the **X** characters from the text	It was a lovely night, so war**X**m that he threw his coat over his arm and did not even put his silk scarf round his throat. As he stro**X**lled home, smoking his. . .	It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his. . .
2	Delete word		It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf **XXXXXX** round his throat. As he strolled home, smoking his. . .	It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his. . .
3	Delete phrase		It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he **XXXXX XXXXX** strolled home, smoking his. . .	It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his. . .
4	Move word (cut - paste)	Move the highlighted words in the correct position (shown as a vertical bar)	It was a lovely night, so warm that he threw his coat over his arm and did not▌ even **put** his silk scarf round his throat. As he strolled home, smoking his. . .	It was a lovely night, so warm that he threw his coat over his arm and did not put even his silk scarf round his throat. As he strolled home, smoking his. . .
5	Move word (select - move)		It was a lovely night, so warm that he threw his coat over his arm and did not▌ even **put** his silk scarf round his throat. As he strolled home, smoking his. . .	It was a lovely night, so warm that he threw his coat over his arm and did not put even his silk scarf round his throat. As he strolled home, smoking his. . .
6	Move phrase	Move the highlighted phrases in the correct position	He **of them** heard one▌ whisper to the other, “That is Dorian Gray.” He **used to be when** remembered how pleased he▌ he was pointed out, or stared at, or talked about. . .	He heard one of them whisper to the other, “That is Dorian Gray.” He remembered how pleased he used to be when he was pointed out, or stared at, or talked about. . .
7	Text correction	Fix the text	It was a lovely night, so war**X**m that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his cigarette, two young men in evening dress passed him. He heard one of them whisper to the other, “That **XXXX** is Dorian Gray.” He remembered how pleased he used to be when he was pointed out, or stared at, or talked about. He was tired of hearing his own name now. Half the charm of the▌ village where he had **little** been so often lately was that no one knew who he was.	It was a lovely night, so warm that he threw his coat over his arm and did not even put his silk scarf round his throat. As he strolled home, smoking his cigarette, two young men in evening dress passed him. He heard one of them whisper to the other, “That is Dorian Gray.” He remembered how pleased he used to be when he was pointed out, or stared at, or talked about. He was tired of hearing his own name now. Half the charm of the little village where he had been so often lately was that no one knew who he was.

Table 3: Editing task list (in order to reduce table size, the task texts are only partially shown).

The given tasks are shown in Table 3. They were designed using [15] as the base, and considering the most significant task for the main editing operations that can be performed (at least partially) with gestures using our technique. The tasks can be divided into two main groups, depending on the predominant type of operation within them (in the gestural editing mode). The first group is formed by delete-intensive tasks (1, 2, 3); the second one is composed of move-intensive tasks (4, 5, 6), while task 7 is a mixed task. The first group can be performed with direct gestures: it is, in fact, possible to perform a single gesture to complete the operation. The second group, instead, can only be performed with multiple gestures.

After the end of the experiment, participants were asked to fill out a System Usability Scale (SUS) questionnaire [5] for each of the two editing techniques. The SUS questionnaire consists of 10 statements to which the user assigns a score on a scale from 1 (strongly disagree) to 5 (strongly agree). The final score of the SUS ranges from 0 to 100. A higher score indicates a greater user usability. Moreover, after the experiment, user free form comments and suggestion were also collected. In particular, at the end of the questionnaire, there was a blank space where each participant could write his/her comments and suggestions about the techniques and a checkbox to state if s/he would like to use the technique in everyday life.

5.4 Design

The experiment was a two-factors within-subjects factorial design. The factors were the text font size (4.0, 5.5 and 7.0 mm) and the editing technique (gesture, traditional). The font sizes were selected by considering 5.5 mm a comfortable size, and adding two more dimensions (one greater by 1.5 mm and one smaller by the same extent). The order of editing technique and font size was counterbalanced between participants, as shown in Table 4.

The dependent variables were the overall task completion time (given by the sum of the 7 task completion times) and the number of failed tasks.

To evaluate user satisfaction the SUS was used for both editing techniques.

Participant	*Font size order*		*Editing technique order*
	Training	*Blocks*	
1	m	s-m-l	gesture - classic
2	m	s-l-m	gesture - classic
3	m	l-s-m	gesture - classic
4	m	l-m-s	gesture - classic
5	m	m-s-l	gesture - classic
6	m	m-l-s	gesture - classic
7	m	s-m-l	classic - gesture
8	m	s-l-m	classic - gesture
9	m	l-s-m	classic - gesture
10	m	l-m-s	classic - gesture
11	m	m-s-l	classic - gesture
12	m	m-l-s	classic - gesture

Table 4: Counterbalancing used during the experiment. Font size abbreviated (s - small, m - medium, l - large).

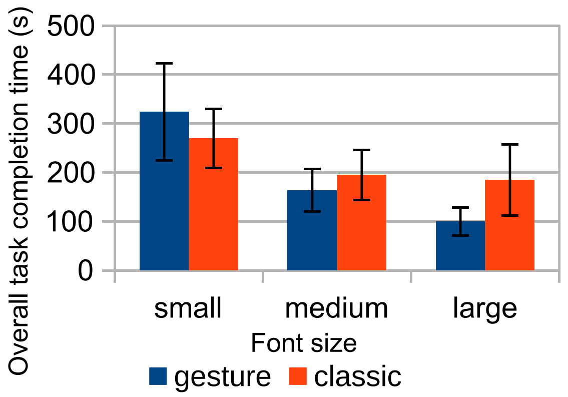


Figure 1: Mean overall task completion time for the two editing techniques, grouped by font size. Error bars show the standard deviation.

6 Results

All participants completed the experiment. The experiment took each of them about half an hour. We tested for significance using a repeated measures analysis of variance (ANOVA). For significant main effects, we used Bonferroni-Dunn post-hoc tests. The alpha level was set to 0.05.

The overall task completion times grouped by font size are shown in Figure 1. As it can be seen, for medium and large font sizes the users were faster with the gestural editing technique (163” vs 195” for medium size, 100” vs 185” for large size), while for small font size they were faster with the traditional technique (324” vs 270”). This is due to the fact that the small font requires greater precision when performing the gestures, and with the capacitive stylus, it is not always easy to work on a small amount of space. Globally the gesture method is sightly faster (196” vs 216”).

From the ANOVA resulted that there was no significant effect of the editing technique ($F_{1,11} = 2.258$, $p = .1611$). The main effect of the font size on the overall task completion time was highly significant ($F_{2,22} = 68.682$, $p < .0001$). The interaction effect between the two factors was also statistically significant ($F_{2,22} = 10.887$, $p = .0005$). A Bonferroni-Dunn post-hoc test revealed significant differences between Gesture-Large and Classic-Large (while no significance was sought between Gesture-Small and Classic-Small, Gesture-medium and Classic-Medium).

The mean completion times for each task are shown in Figure 2. We report a separate chart for small (a), medium (b) and large (c) font and for their aggregated values (d). The gestural editing technique was almost always faster than the classic technique for the medium and large font sizes, with task 6 (*move phrase*) as the only exception for the medium font. The classic technique was instead almost always faster with small font, with the exception of task 2 (*delete word*) and task 3 (*delete phrase*).

Finally, it turned out that the participants failed fewer tasks with the gestural editing technique compared to the traditional editing technique. In particular, only one failed task occurred for the gestural technique (with small font size), while 6 failed tasks occurred for the traditional technique (4 with small font size and 2 with large font size). Nevertheless, the ANOVA results show no significant effects for the editing technique ($F_{1,11} = 2.570$, $p = .1372$), the font size ($F_{2,22} = 2.714$, $p = .0884$), or their interaction ($F_{2,22} = 0.865$, $p = .4348$).

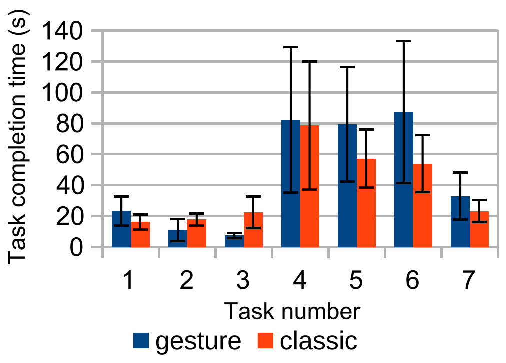
6.1 User Satisfaction and Free-form Comments

The average SUS score was 61.25 for the classic editing technique ($SD = 18.68$) and 73.13 ($SD = 15.88$) for the gestural editing technique. A Wilcoxon matched-pairs signed-ranks test performed on SUS scores revealed a statistical significance between the two techniques ($Z = -2.138$, $p < .05$).

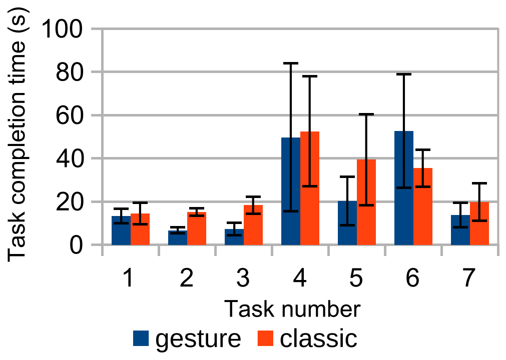
When asked which editing technique they prefer, all participant chose the gestural one, highlighting a greater ease in task execution with the possibility of omitting many actions. The only problem highlighted by all participants was the lack of usability with the small font size. Some users suggested fusing the two techniques in order to allow greater efficiency. The action that proved to be more complex for participants were those involving text moving, which requires a greater effort in some conditions. Regarding the traditional technique, some participants pointed out that it would be appropriate to add an *Undo* button, to limit the cases in which it is necessary to restart a task.

6.2 Discussion

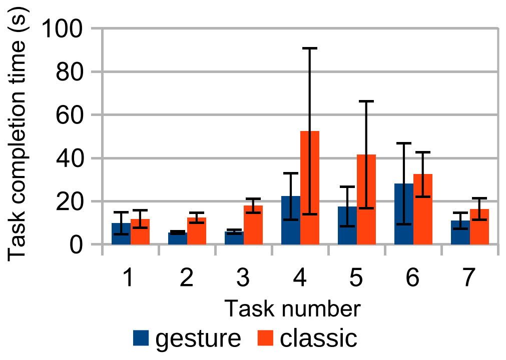
During the experiment, the participants could consult the sheet with the gesture set supported by the gestural editing mode, while in a real context this guide would not be available. A further study will be needed to understand how easy it is for the users to learn the gesture set and how useful the addition of an interactive help would be. However, from participants' comments and results, the gesture set seemed quite easy to learn and use, as expected since their choice is based on the results of the preliminary study. There was no consensus among participants in choosing the best ges-



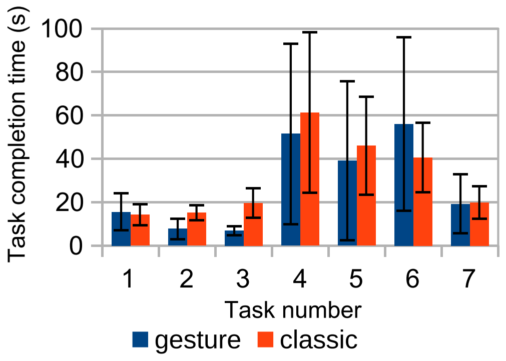
(a) Small font mean task completion times.



(b) Medium font mean task completion times.



(c) Large font mean task completion times.



(d) Mean task completion times (mean for all font sizes).

Figure 2: Task completion times. Error bars show the standard deviation.

ture sequence to move text: about 50% of them preferred to cut-paste while the other 50% preferred select-move. This reflects their preference when operating on a touch device. However, on average, select-move is faster than cut-paste, as can be seen by looking at tasks 4 and 5 in Figure 2.

Some participants also complained about cursor positioning. Especially with small font size, it was difficult for them to place the cursor quickly. This is a general issue found in text editing applications, and some techniques are used to improve the placing, as described in Section 2. Some participants suggested adding one of those techniques in order to allow a further efficiency increase. When assessing the results one must also consider that the participants already knew the traditional editing technique, while they had to learn the new gesture based technique. Despite this, all participants showed a fast learning process.

7 Conclusions and further works

This paper presents a new gesture-based text editing technique, which allows the user to perform operations such as text deletion and moving text more efficiently. The technique was designed by taking into account the most natural gestures for users, investigated through a preliminary study. A user study was also conducted to compare the proposed technique with the classical one. The results show that the gestural editing technique outperforms the traditional one when the text font size increases. The feedback about the gestural technique is positive, and participants showed that the gestures can be learned in a short time. Future work includes a refinement of the gesture set in order to allow greater user accuracy, particularly on small fonts. The participants' proposal to integrate a technique to facilitate cursor placement with small fonts can also be implemented. Lastly, the suggestions of fusing the gesture and classical techniques might also be considered, in order to allow increased user satisfaction and performance.

Future studies will also aim at mitigating the threats to the validity of this study, such as increasing the number of participants and testing with different device and stylus types.

8 Acknowledgment

The authors thank Luigi Dell'Aglio, Domenico Desiato, Giuseppe Pietravalle and Giuseppe Santaniello for their support in carrying out the experiments.

References

[1] C. Alvarado and R. Davis. Sketchread: A multi-domain sketch recognition engine. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, pages 23–32, New York, NY, USA, 2004. ACM.

[2] C. Appert and S. Zhai. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 2289–2298, New York, NY, USA, 2009. ACM.

[3] N. S. Borenstein. The evaluation of text editors: A critical review of the roberts and morgan methodology based on new experiments. *SIGCHI Bull.*, 16(4):99–105, Apr. 1985.

[4] A. Bragdon, E. Nelson, Y. Li, and K. Hinckley. Experimental analysis of touch-screen gesture designs in mobile environments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 403–412, New York, NY, USA, 2011. ACM.

[5] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[6] W. Buxton, E. Fiume, R. Hill, A. Lee, and C. Woo. Continuous hand-gesture driven input. In *Graphics Interface*, volume 83, pages 191–195, 1983.

[7] C. Chen, S. T. Perrault, S. Zhao, and W. T. Ooi. Bezelcopy: An efficient cross-application copy-paste technique for touchscreen smartphones. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, AVI '14, pages 185–192, New York, NY, USA, 2014. ACM.

[8] M. L. Coleman. Text editing on a graphic display device using hand-drawn proofreader's symbols. In *Pertinent Concepts in Computer Graphics, Proceedings of the Second University of Illinois Conference on Computer Graphics*, pages 283–290, 1969.

[9] G. Costagliola, M. De Rosa, and V. Fuccella. Local context-based recognition of sketched diagrams. *Journal of Visual Languages & Computing*, 25(6):955 – 962, 2014.

[10] G. Costagliola, M. De Rosa, and V. Fuccella. Extending local context-based specifications of visual languages. *Journal of Visual Languages & Computing*, 31, Part B:184 – 195, 2015.

[11] G. Costagliola, V. Fuccella, and M. D. Capua. Interpretation of strokes in radial menus: The case of the keyscretch text entry method. *Journal of Visual Languages & Computing*, 24(4):234 – 247, 2013.

[12] L. Findlater, B. Q. Lee, and J. O. Wobbrock. Beyond qwerty: augmenting touch-screen keyboards with multi-touch gestures for non-alphanumeric input. In *Proc. CHI'12*, pages 2679–2682, 2012.

[13] V. Fuccella and G. Costagliola. Unistroke gesture recognition through polyline approximation and alignment. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3351–3354, New York, NY, USA, 2015. ACM.

[14] V. Fuccella, M. De Rosa, and G. Costagliola. Novice and expert performance of keyscretch: A gesture-based text entry method for touch-screens. *IEEE Transactions on Human-Machine Systems*, 44(4):511–523, Aug 2014.

[15] V. Fuccella, P. Isokoski, and B. Martin. Gestures and widgets: Performance in text editing on multi-touch capable mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI 2013, pages 1–10, New York, NY, USA, 2013. ACM.

[16] T. Hammond and R. Davis. Ladder, a sketching language for user interface developers. *Computers & Graphics*, 29(4):518 – 532, 2005.

[17] A. Lee and F. H. Lochovsky. Enhancing the usability of an office information system through direct manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '83, pages 130–134, New York, NY, USA, 1983. ACM.

[18] L. A. Leiva, V. Alabau, V. Romero, A. H. Toselli, and E. Vidal. Context-aware gestures for mixed-initiative text editing uis. *Interacting with Computers*, 27(6):675, 2015.

[19] I. S. MacKenzie and R. W. Soukoreff. Text entry for mobile computing: Models and methods,theory and practice. *Human–Computer Interaction*, 17(2-3):147–198, 2002.

[20] T. L. Roberts. *Evaluation of computer text editors*. PhD thesis, Stanford University, Stanford, CA, USA, 1980.

[21] T. L. Roberts and T. P. Moran. The evaluation of text editors: Methodology and empirical results. *Commun. ACM*, 26(4):265–283, Apr. 1983.

[22] D. Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25(4):329–337, July 1991.

[23] J.-B. Scheibel, C. Pierson, B. Martin, N. Godard, V. Fuccella, and P. Isokoski. Virtual stick in caret positioning on touch screens. In *Proceedings of the 25th Conference on L'Interaction Homme-Machine*, IHM '13, pages 107:107–107:114, New York, NY, USA, 2013. ACM.

[24] C. G. Wolf and P. Morrel-Samuels. The use of hand-drawn gestures for text editing. *International Journal of Man-Machine Studies*, 27(1):91 – 102, 1987.

Improving MapReduce Performance by Using a New Partitioner in YARN

Wei Lu[1,], Lei Chen[1,*], Haitao Yuan[1], Weiwei Xing[1], Liqiang Wang[2], Yong Yang[1]
[1] School of Software Engineering,Beijing Jiaotong University, Beijing, China
[2] Department of Computer Science, University of Central Florida, Orlando, USA
Email: [1]{luwei,13112084,htyuan,wwxing,12112088}@bjtu.edu.cn
[2]{lwang}@cs.ucf.edu

Abstract

Data skew, cluster heterogeneity, and network traffic are three issues that significantly influence the performance of MapReduce applications. However, the Hash-Partitioner in native Hadoop does not consider them. This paper proposes a new partitioner in Yarn (Hadoop 2.6.0), namely, PIY, which adopts an innovative parallel sampling method to achieve the distribution of the intermediate data. Based on this, firstly, PIY mitigates data skew in MapReduce applications. Secondly, PIY considers the heterogeneity of the computing resource to balance the load among Reducers. Thirdly, PIY reduces the network traffic in shuffle phase by trying to retain intermediate data on those nodes who act as both mapper and reducer. Compared with the native Hadoop and some other popular strategies, PIY can reduce the execution time by 35.62% *and* 50.65% *in homogeneous and heterogeneous cluster, respectively. We also implement PIY in parallel image processing. Compared with several existing strategies, PIY can reduce the execution time by* 11.2%

MapReduce; Hadoop; data skew; load balance; data transmission amount; heterogeneousparallel image processing

1 Introduction

MapReduce has been proven to be an effective tool to process large data sets [10]. As a parallel computing framework that supports MapReduce, Apache Hadoop [6] is widely used in many different fields. MapReduce consists of two main functions: the map function, which transforms input data into intermediate data, namely <key,value> pairs, and the reduce function, which is applied to list of values that correspond to the same key. Partitioning[4] is a critical feature of MapReduce because it determines the reducer to which an intermediate data item will be sent in shuffle phase. Hadoop 2.6.0 usually employ static hash functions to partition the intermediate data, which is called Hash-Partitioner and described as the formula (1). Although MapReduce is currently gaining wide popularity in parallel data processing, its Hash-Partitioner is still inefficient and has room for improvement.

$$Hash(Hashcode(Intermediate\,data)\,mod\,ReducerNum) \tag{1}$$

First, data skew [2] is one of the most serious impact factors affecting the performance of Hadoop cluster. Data skew refers to the imbalance in terms of data allocated to each task or the imbalance in terms of work required to process such data. When data skew occurs, the aforementioned Hash-Partitioner leads to the fact that most of nodes have to remain idle after they complete their tasks and await the stragglers. Finally this approach prolongs the execution time and decreases the computing efficiency. Therefore, balancing the hash partition size, which is defined as the size of the key-value pairs with the same hash result, is an important indicator for load balancing among the reducers.

Secondly, heterogeneity is neglected by the Hash-Partitioner. The computing environments for MapReduce in the real world always are heterogeneous [17]. Even if data skew does not happens on intermediate data, the execution time of each node are diverse because their various computing capacities, consequently, stragglers still exist in clusters. Therefore, Hash-Partitioner can not work well in heterogeneous Hadoop cluster.

Thirdly, with the increasing size of computing clusters, it is common that many nodes act as both Mapper and Reducer in real production environment. Obviously, the more intermediate data stay in these nodes, the less network traffic happen in shuffle phase[20]. However, the Hash-Partitioner does not consider this fact.

Many studies have focused on the data skew mitigation. Among the proposed solutions, some are specific to a particular type of applications [9][13], some require a pre-sample of the input data [18][12][16], some identify the task with

DOI reference number: 10.18293/DMSVLSS2017-002

the greatest expected remaining processing time and repartitions the unresolved data in a way that fully utilizes the nodes in the cluster [10]. There are lot of studies on the heterogeneous Hadoop cluster [19] to reduce network traffic in shuffle phase [11]. However, all previous studies can not well solve the three deficiencies mentioned above comprehensively.

This paper proposes a new partitioner for Yarn (Hadoop 2.6.0), namely, PIY, to solve the problems about data skew and network traffic in shuffle phase in heterogeneous Hadoop cluster. Compared with the previous studies, the contributions of this paper can be summarized as follows:
(1) We propose a novel sampling method, named PRS. PRS achieves a highly accurate approximation to the distribution of the intermediate data by parallelly sampling the input data during the normal map processing, and it only causes little overhead.
(2) We propose an algorithm, namely BASH, to tackle the data skew problem.
(3) To avoid the degradation performance caused by heterogeneity, PIY allocates appropriate amount of intermediate data to reducers according to their computing capacity.
(4) PIY optimizes the network traffic by decreasing the amount of the transmitted data located on nodes acting as both Mapper and Reducers.
(5) We conduct a performance evaluation with PIY in YARN (Hadoop 2.6.0). Compared with some other popular strategies, PIY can reduce the execution time by 35.62% and 50.65% in homogeneous and heterogeneous Hadoop cluster, respectively. We also implement PIY in parallel image processing. Compared with several existing strategies, PIY can reduce the execution time by 11.2%

The rest of this paper is organized as follows. Section 2 reviews related studies. Section 3 briefly introduces the $Approx\_Subset\_Sum$ algorithm, which is used by PIY. Section 3 describes our PIY in detail. Section 5 describes the performance evaluation of PIY. Finally, Section 6 concludes this paper.

2 Related Work

To ascertain the distribution of the intermediate result before determining the partition in Hadoop, sampling methods are widely applied in previous studies. We classify these methods into two categories. The first category is to launch a pre-run extra job before whole normal jobs to conduct data distribution statistics, and then decide an appropriate partition [18]. The drawback of these methods is that when the volume of data is large, sampling will cost much time which results in prolonging the execution time of whole job. The second category contains the methods that integrate sampling into the map stage [2]. However, these methods hardly achieve high sampling accuracy, and also cause performance degradation because the parallel degree is decreased between the map and the reduce stage.

Data skew has also been studied in the MapReduce environment during the past few years. In [6], Ibrahim et al. proposed LEEN, which partitions all intermediate keys according to their frequencies and the fairness of the expected data distribution after the shuffle phase. However, LEEN lacks preprocessing to estimate the data distribution effectively and separates map and reduce tasks absolutely, and therefore, it incurs significant time cost. Gufler et al. proposed TopCluster [3], which can mitigate data skew among reducers by estimating the cost of each intermediate partition. However, it increases the intermediate data transmission amount in shuffle because it ignores data locality in reduce side.

Heterogeneous computing environment is a research hotspot in recent years. LATE [19] calculates the progress rate of tasks and selects the slow task with the longest remaining time to back up. The work in [17] presents a system that adopts the virtualization technology to allocate data center resources dynamically based on application demands. Their common limitation is that they cannot solve the data skew problem.

However, all the aforementioned approaches ignore the fact that there are plenty of nodes that run map tasks and Reduce tasks concurrently in large-scale computing cluster. The network traffic in shuffle phase will be optimized obviously if the partitioner can reduce the transmission amount of the intermediate data that stay on those nodes. Our approach, i.e., PIY, can comprehensively resolve all problems mentioned in this section.

3 Approx-Subset-Sum Algorithm

Because our PIY algorithm is based on the Approx-Subset-Sum algorithm[8], we introduce it in this section. An instance of the subset-sum problem is a pair (L, t), where *L* is a set $x_1, x_2, ..., x_n$ of *n* positive integers (in arbitrary order) and *t* is a positive integer. This decision problem is to find whether there exists a subset of *L* that adds up exactly to the target value *t*. As we known, this problem is NP-complete and traversing all subset of *L* will take exponential time, and therefore, this is unacceptable when the data being processed is extremely large. To reduce the time complexity, the Approx-Subset-Sum algorithm trims list *L* by selecting and remaining only one value *Z* to represent all the values *Y* according to the formula (2) and finally get the list $L^{'}$. Here ε $(0 < \varepsilon < 1)$ is a trimming parameter. We assume there is a *Z* that represents *y* in the new list $L^{'}$. Each removed element *y* is represented by a remaining element z that satisfies formula (2). Obviously, trimming can dramatically decrease the number of elements kept while remaining a close (and slightly smaller) representative value in the list for each deleted element. Algorithm 1 describes the procedure of trimming list *L* that contains m elements in time $\Theta(m)$. It is assumed that L is sorted in monotonically increasing order. The output of the procedure is a trimmed and sorted list.

$$\frac{Y}{1+\varepsilon} \leq Z \leq Y \tag{2}$$

Algorithm 1 Trim Algorithm

```
Input:  L: a positive integer set contains m factors $< l_0, ..., l_{m-1} >$; $\varepsilon$: trimming parameter;
Output: $L^{'}$
 1: m = L.length;
 2: $L^{'} = \langle l_0 \rangle$;
 3: last = $l_0$;
 4: for i = 1 to m-1 do
 5:     if $l_i > last * (1+\varepsilon)$ then
 6:         append $i_1$ $onto\ the\ end\ of\ L^{'}$;
 7:         last = $l_i$;
 8:     end if
 9: end for
10: return $L^{'}$;
```

Algorithm 2 Approx-Subset-Sum

```
Input:  S: a positive integer set contains n elements $< S_0, ..., S_{n-1} >$; L: a positive integer set; t: target value; $\varepsilon$(0<$\varepsilon$<1): trimming parameter; $L_i$: the generated list after the $S_i$ is appended.
Output: $z^*$: the largest value in $L_n$
 1: n = the length of L
 2: $L_0$ = $< 0 >$
 3: for i=0 to n-1 do
 4:     $L_i$ = Merge-Lists($L_{i-1}$, $L_{i-1} + S_i$)
 5:     $L_i$ = Trim($L_i$, $\varepsilon$/2n)
 6:     remove every element that is greater than t from $L_i$.
 7: end for
 8: return $z^*$
```

The Approx-Subset-Sum algorithm is described as Algorithm 2. It returns a value z^* whose value is within a 1+ε factor of the optimal solution. Line 2 initializes the list L_0 to be the list containing just the element 0. For loop in lines 3 - 6 computes L_i as a sorted list containing a suitably trimmed version of the set L_{i-1} , with all elements larger than t removed. MERGE-LISTS(L, $L^{'}$) in line 4 returns the sorted list that is the merge of its two sorted input lists L and $L^{'}$ with duplicate values removed. L_{i-1} + S_i denotes the list of integers derived from L_{i-1} by increasing each element of L_{i-1} by S_i. For example, if L_{i-1} = $\langle 1, 2, 3, 5, 9 \rangle$, then L_i = L_{i-1} + 2 = $\langle 3, 4, 5, 7, 11 \rangle$. Trim($L_i$, ε/2n) in line 5 decreases the length of L_i with the trimming parameter ε/2n.

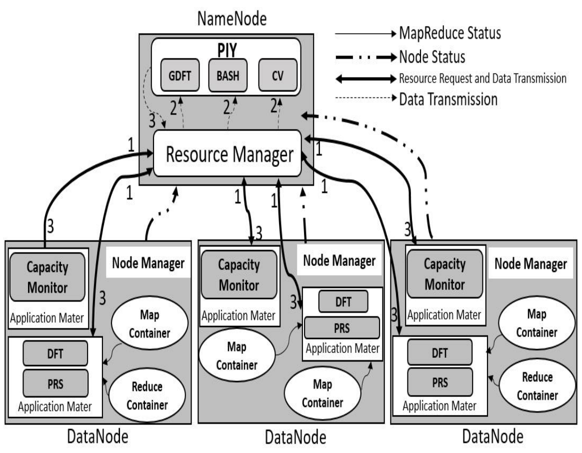


Figure 1: The Architecture of PIY

Here is an example to illustrate the execution of Approx-Subset-Sum algorithm. It is assumed that the instance S=$\langle 104, 102, 201, 101 \rangle$, t=308 and ε=0.40. The trimming parameter is $\varepsilon/8 = 0.05$. $Approx\_Subset\_Sum$ computes the following values in the indicated lines:
line 2: $L_0 = \langle 0 \rangle$
line 4: $L_1 = \langle 0, 104 \rangle$
line 5: $L_1 = \langle 0, 104 \rangle$
line 6: $L_1 = \langle 0, 104 \rangle$

line 4: $L_2 = \langle 0, 102, 104, 206 \rangle$
line 5: $L_2 = \langle 0, 102, 206 \rangle$
line 6: $L_2 = \langle 0, 102, 206 \rangle$

line 4: $L_3 = \langle 0, 102, 201, 206, 303, 407 \rangle$
line 5: $L_3 = \langle 0, 102, 201, 303, 407 \rangle$
line 6: $L_3 = \langle 0, 102, 201, 303 \rangle$

line 4: $L_4 = \langle 0, 101, 102, 201, 203, 302, 303, 404 \rangle$
line 5: $L_4 = \langle 0, 101, 201, 302, 404 \rangle$
line 6: $L_4 = \langle 0, 101, 201, 302 \rangle$

The algorithm returns z^* = 302 as its answer, which is maintaining within 2% of the optimal answer 307 = 104 + 102 + 101. This shows that the $Approx\_Subset\_Sum$ algorithm can find an approx optimal solution in a fully polynomial time, therefore, the overhead it causes is acceptable.

4 A New Partitioner in Yarn

4.1 System Overview

We designed a new partitioner in Yarn, named PIY, based on Hadoop 2.6.0, and the architecture of PIY frame is shown in Figure 1. In particular, each Parallel Reservoir Sampler (PRS) samples the input data on each Mapper.

The Data Frequency Table (DFT) creates a table that records the value of each key in each DataNode according to the sampling statistics. The Capacity Monitor fetches the computing capacity value of each DataNode. The Global DFT (GDFT) summarizes all DFT data in each DataNode. The CV records the computing capacity values of all DataNodes. The BASH is the core unit in our PIY that generates the final partitioning result. The workflow of PIY consists of 3 steps:

(1) When split operation in map stage finishes, PRS applies to Resource Manager for containers to conduct sampling. All the sampled <key,value> pairs are summarized and stored into a file. The detailed process of PRS is described in the following Section 4.2. The DFT counts and records the sampled <key,value> pairs in each DataNode and generates a key frequency table. Here the key frequency refers to the number of pairs corresponding to each key. At the same time, the Capacity Monitor collects the computing capacity of each DataNode, which is described in Section 4.3. When all these processes are completed, the information data, which consists of the key frequencies in DFT and the computing capacity value of each DataNode will be transmitted from the Application Master to the Resource Manager through heartbeat messages.

(2) When the Resource Manager receives the information data, it will transmit corresponding data to the GDFT and CV, respectively. Then the GDFT summarizes all the key frequencies in each DataNode into a total frequency table. The CV records the computing capacity values of all DataNodes by the data from the Capacity Monitor. These are essential preparative works for the final partitioning results generated by the unit BASH.

(3) Then the BASH generates the final partitioning result using the algorithm described in Section 4.5. Finally, the Resource Manager will transmit the results back to the application Masters through the resource response message.

4.2 Parallel Reservoir Sampling Strategy

In PIY, we get the distribution of the intermediate result by running a novel sampling during the normal map processing. Our sampling is performed by some map tasks with higher priority. Therefore, when split operation in map stage finishes, the map tasks conducting sampling are processed preferentially. Obviously, there is tradeoff between the sampling overhead and the accuracy of the result. In our experiments, we find that by integrating sampling into 20% of map tasks, a sufficient accuracy approximation can be achieved. Our sampling strategy, namely Parallel Reservoir Sampling, simply PRS for short, is based on reservoir sampling algorithm[15]. PRS runs by invoking the class org.apache.hadoop.mapreduce.lib.InputSampler and overloading the SplitSampler method.

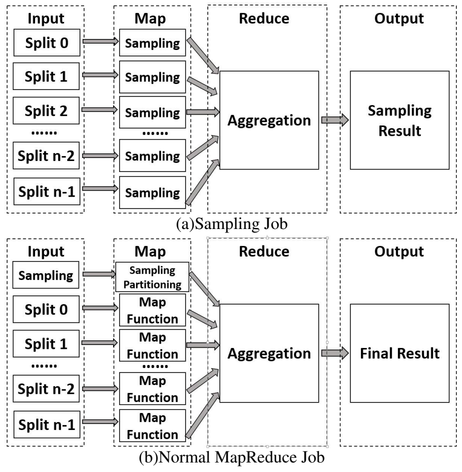


Figure 2: The Process of Parallel Reservoir Sampling

The main idea of PRS is described as follows. PRS builds one reservoir for each split and samples K elements from it. All key/value pairs in each split are scanned and the first K elements are stored in each reservoir. For a key/value pair whose sequence number is larger than *K*, we replace stored elements with it based on a certain probability. This process is executed for each reservoir in parallel. All the sampled key/value pairs are summarized together and stored into a file by the reduce function. The process of PRS is shown as Figure 2(a). When the sampling tasks of some splits are finished, their corresponding normal user-defined map function are executed successively. Thus, our PRS is integrated into the normal map processing. As is shown in Figure 2(b), when all samplers are complete, the sampling result will be aggregated and transmitted to GDFT model in the NameNode which decides the sampling partition. In our system, Reducers begin to pull their input data after the sampling partition is decided. This is later than the default start time of reduce stage in native hadoop because the decision of sampling partition introduces overhead. However, this overhead is negligible based on our experimental results shown in Section 5.

4.3 The Capacity Monitor

The Hadoop cluster is ofen heterogenous. To get the computing capacity of each DataNode, we designed a special model, namely, Capacity Monitor, in each DataNode. To decrease the extra overhead dramatically, the Capacity Monitor in each DataNode keeps monitoring the implementation of the sampled input data when the sampling function begins to run, and gets its consuming volume $Volume(con)_{id}(1 \leq id \leq m)$ during a period of time Δt, here *m* denotes the number of DataNode in cluster. Then we can calculate the capacity value of the id^{th} DataNode, CV_{id}, by following formula (3). Capacity Monitors sends the capacity values of the DataNode to the PIY in NameNode through the heartbeat message.

$$CV_{id} = Volume(cons)_{id}/\Delta t \tag{3}$$

4.4 Network Traffic in Shuffle Phrase

As the bandwidth is the scarce resource in networks, the shuffle phase has become the bottleneck of MapReduce due to its large amount of network traffic. As is known, there are many Reducers in cluster, especially in Datacenter Network (DCN). In addition, many DataNodes act as both Mapper and Reducer[12]. If we can stay as many as intermediate <key,value> pairs on these DataNodes by the partition method in shuffle phase, it also furthest decrease the network traffic[5]. It's assumed that there are many <key,value> pairs corresponding to a special key on those DataNodes simultaneously. The BASH algorithm will find the DataNode that contains the maximum amount of these pairs, and then transmits all the pairs corresponding to the key to this DataNode. Our experimental result proves this method could decrease the network traffic in shuffle phase by up to 19.11%.

4.5 BASH Algorithm

In this section, we describe our proposed algorithm named BASH that comprehensively considers the load **BA**lance among all Reducers, network traffic in **S**huffle and the **H**eterogeneity of Hadoop cluster. As shown in algorithm 3, there are three steps in BASH. First, it minimizes intermediate data transmission in shuffle phase. Second, it gets the data volume that each reducer should process according to their computing capacity. Finally, to balance the load among Reducers, BASH partitions intermediate data to each Reducer using Approx-Subset-Sum algorithm.

We assume that there are *k* distinct <key,value> pairs corresponding to various keys, and *r* Reducers in cluster. key_dest_i records the serial number of the destination Reducer that will process the <key,value> pairs corresponding to key_i, and all key_dest_i consist of the array key_dest[1,...,k]. Lines 1-3 initialize all key_dest_i with -1, which means all <key,value> have not been partitioned. The array RS[1,...,r] records the volume of data that should be processed by special Reducers. CV[1,...,r] records the computing capacity of each Reducer, the value of CV_i can be obtained by formula (3). Sum_CV records the total computing capacity value of all Reducers. Lines 4-7 initialize all RS_i 1≤i≤r, and compute the Sum_CV.

Lines 8-17 reduce the amount of network traffic in shuffle phase. As described in section 4.4, we focus on the

Algorithm 3 BASH Algorithm

Input: **k**: the number of <key, value> pairs; **r**: the number of Reducer; **key_size[1,..,k]** : the data volume of all <key,value> pairs ; **CV[1,...,r]** : the computing capacity value of every Reducer; **Sum_CV**: the total capacity value of all Reducers; ε : approximation parameter; **RS[1,..,r]**: the volume of the data that have been determined to be processed in every Reducer; **T[1,...,r]**: the remaining capacity of every Reducer; **Total_Size**: the total volume of all <key,value> pairs produced by all Mappers.

Output: **key_dest[1,...,k]**: A array indicating the destination Reducer of every key;

```
1:  for i = 1 to k do
2:      $key\_dest_i = -1$;
3:  end for
4:  for i = 1 to r do
5:      $RS_i = 0$;
6:      $Sum\_CV = Sum\_CV + CV_j$;
7:  end for
8:  for each Reducer $R_j (1 \leq j \leq r)$ do
9:      if $R_j$ is also a Mapper then
10:         for every $key_i$ on $R_j$ do
11:             if key_$dest_i$ == -1 then
12:                 $key\_dest_i$ = $MaxReducer(key_i)$;
13:                 $RS_{key\_dest_i} = RS_{key\_dest_i} + key\_size_i$;
14:             end if
15:         end for
16:     end if
17: end for
18: for j = 1 to r do
19:     $T_j = Total\_Size * (CV_j/Sum\_CV) - RS_j$;
20: end for
21: for j=1 to r do
22:     $Z^*$ = $Approx\_Subset\_Sum(key\_size[1,...,k]$,$T_j, \varepsilon)$;
23:     Set $key\_dest$ of the keys which composing $Z^*$ to
        the sequence number of $Reducer_j$;
24: end for
25: for i = 1 to k do
26:     if $key\_dest_i == -1$ then
27:         $key\_dest_i$ = the sequence number of the
        strongest capacity Reducer;
28:     end if
29: end for
30: return $key\_dest[1,...,k]$;
```

DataNodes who act as both Mapper and Reducer. For each key_i ($1 < i < k$) on these Reducers, BASH first checks whether its destination Reducer is determined. If not, the function MaxReducer(key_i) in Line 12 will find the Reducer on which the volume of the <key,value> pairs corresponding to the key_i is the maximum, and then set this Reducer as the destination Reducer of key_i. Line 13 updates the volume of the data that should be processed on this Reducer. Here, the array key_size[1,...,k] records the data volume of all <key,value> pairs. Lines 18-20 get the remaining capacity of each Reducer, which is denoted as array *T*. Here remaining capacity means the extra data volume that one Reducer can process. The Total_Size denotes the total volume of experimental data set. Total_Size * (CV_j/Sum_CV) means the total data volume that the $Reducer_j$ should process according to its computing capacity.

Using Approx_Subset_Sum algorithm, lines 21-29 balance the load among all Reducers by partitioning the intermediate data based on Reducer's computing capacity. In lines 21-24, BASH partitions intermediate data to each Reducer and records the destination Reducer of each key into the array key_dest. We can get these value through the GDFT in PIY. As we describe in Section 3, the Approx_Subset_Sum algorithm only gets an approximate result that does not reach the target value. Therefore, the amount of data that is partitioned to each Reducer could not reach its capacity value. This generates some trivial datasets that are not partitioned finally. In lines 25-29, BASH assigns these trivial datasets to the Reducer with the strongest computing capacity.

5 Evaluation

In this section, we describe the performance evaluation of PIY by running two popular benchmarks with synthetic and real-world data sets whose data skew rate are different, our experiments are performed under both homogeneous and heterogeneous environments. Specially, we evaluate PIY to process large-sized imagine in parallel.

5.1 Experimental Environment

In our experiments, we set up two Hadoop clusters, one is homogeneous, and the other is heterogeneous. Our Hadoop homogeneous cluster consists of 60 physical machines installed with Ubuntu 12.04(KVM as the hypervisor) with 16 core 2.53GHz Intel processors, 4G memory, and the 60 nodes connected through a single switch, the network bandwidth is 1Gbps. Our experiments are performed in YARN (Hadoop 2.6.0). All nodes are used as both computing and storage nodes. The HDFS block size is set to 64 MB and each node is configured to run at most 6 map tasks and 2 reduce tasks concurrently. Our heterogeneous cluster contains 60 physical machines with three types. The first type contains 30 machines with 16 core 2.53GHz Intel processors, 4G memory. The second type contains 20 machines with 4 core 3.3GHz Intel processors, and 8G memory. The third type contains 10 machines with 2.4GHz Intel processors, and 2G memory. The other configurations are same in the homogeneous cluster.

Table 1: Jobs with Different Sampling Methods

Sampling Method	Time(s)	Sample File Size(MB)	$Accu_{appro}$
Random	2.8	1.2	307889
TopCluster	2.5	1.3	142728
PRS	2.6	5	97335

In this section, we evaluate PIY by running different type of bench-marks in homogeneous and heterogenous Hadoop cluster, respectively. In order to ensure accuracy, we performed each group of experiments at least 10 times and took the mean value as the final result so as to reduce the influence of the environment.

5.2 Accuracy of the Sampling Method

We compare our PRS with the other two sampling methods: the random sampler used in native Hadoop and TopCluster [3]. We run three different samplers on a 10GB real-word data sets from the full English Wikipedia archive, which contains 50000 keys. We measure the sampling approximation by formula(4), where x_i^{appro} and x_i^{real} denote the sampling and real frequency of tuple corresponding to the key *i*, respectively. The smaller value of $Appro_{sampl}$, the better. All three methods sample 20% of input splits and 1000 keys from each split. Table 1 shows that the size of sample file generated from PRS is larger than the others meanwhile their execution time are approximately equal. This is because PRS completes reservoir sampling on each split in parallel and collects the sample result with larger volume. The better accuracy can be realized if the sampling result is larger.

From Table 1 we can see that the approximation of our PRS is 97335, which is better than the other two sampling methods. This is also visualized in Figure 3 for the top 1000 large keys in the data. Note that the sampling approximation of TopCluster is fairly accurate on the large keys which are at the beginning of the curve, representing their frequency are relatively large, but terrible on the keys whose frequency is lesse than 10^3. The reason is that TopCluster assumes the distribution of small keys are in accord with the large keys, and this assumption can be misleading when there are a large number of small keys in the data.

$$Appro_{sampl} = \sqrt{\frac{\sum_{i=1}^{n} (x_i^{appro} - x_i^{real})^2}{n}} \qquad (4)$$

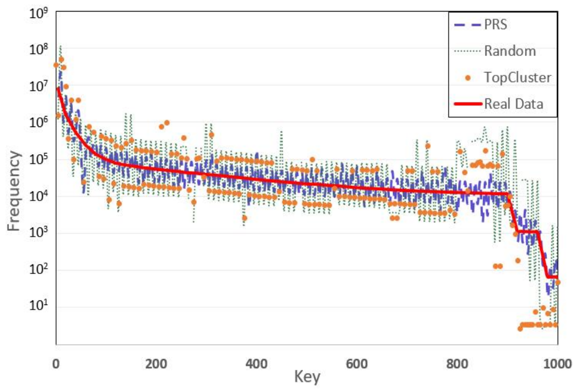


Figure 3: Comparison of three sampling methods in Grep

5.3 Load Balance among Reducers When Running Sort Benchmark

One of majar motivations for PIY is to balance loads among Reducers when data skew happens. Therefore, in this section, we evaluate PIY by running Sort benchmark, which represents reduce-input-heavy job, to process the input data with various data skew degrees. In this paper, the load balancing and data skew degree are measured by the coefficient of variation, which is represented as COV. The smaller COV, the better. Figure 4(a) and Figure 4(b) show COV when running sort benchmark based on 10 GB of synthetic data in homogeneous and heterogenous cluster respectively. We generate a 10GB synthetic data set following Zipf [1] distributions with varying δ parameters from 0.2 to 1.2 to control the degree of the skew.

As Figure 4(a) shows, the curves of Hadoop-Hash and SkewTune keep rising when the data skew rate increases, while the COV of PIY remains very low all the time. This can be explained as PIY partitions the intermediate data to all Reducers evenly in homogeneous cluster. The reason why SkewTune performs worse than PIY is that SkewTune can only repartition the input data of one straggler at a time, it could not balance the loads on all Reducers when there are more than one slow reduce tasks caused by serious skew data. On account of the Hadoop-Hash partitions data by the hash code of keys, it is easy to unbalance loads seriously when data skew happens, which makes it the worst performance in Figure 4 (a).

From Figure 4(b), we find the same results as in Figure 4(a).In addition, while the value of PIY are almost unchange, the values of Hadoop-Hash and SkewTune at most of data skew rates are higher than that in homogeneous cluster, and this trend is more obvious when the data skew rate increases. In other words, the optimization degree of PIY in load balance in heterogeneous cluster is much more than that in homogenous cluster. Beside the reasons we have described in the prior paragraph, the consideration of heterogeneity of PIY made a greater contribution in load balance among Reducers.

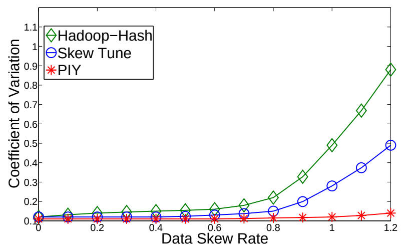
5.4 Execution Time of Sort Benchmark

Figure 4 also shows the execution time of the experiments we have described in Section 5.3. The curves in Figure 4(c) shows the results in the homogenous cluster. We can see PIY is faster than Hadoop-Hash and SkewTune when processing the data with high skew rate. On the contrary, when the data skew rate is lower than a certain threshold, PIY does not perform satisfactorily. The reason is that when the data skew degree is low, e.g. less than 0.28 in our experiment, the Hadoop-Hash has the shortest execution time in the homogeneous cluster because of its even partitions of intermediate data without extra overhead. SkewTune produces small overhead on moving unprocessed data of the slower tasks because there are few stragglers in this scenario, and this leads to its execution performance behinds the Hadoop-Hash. Furthermore, PIY consumes the longest execution time because of its extra overhead produced by sampling data and making partition decision in map phase. However, as data skew degree increases, the optimization, which is achieved through balancing load among Reducers using approx_sum_subset algorithm, gradually offsets the time spent by the extra overhead. Therefore, PIY consumes the lowest execution time. Consequently, compared with Hadoop-Hash and SkewTune, PIY achieves the average improvements in execution time by 16.39% and 4.71%, respectively, and the maximum improvements reach 35.62% and 9.90%, respectively, when the data skew rate is 1.2.

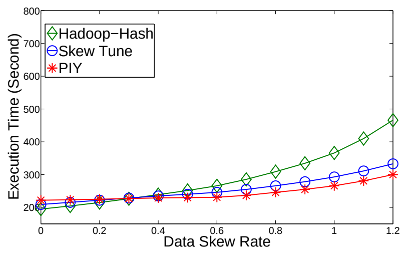
Figure 4(d) shows the fully adaption of PIY to heterogeneous cluster. PIY is also the fastest one in most cases. The threshold, less than which the performance of PIY is worse than the other two, is 0.17 and it less than 0.28 in Figure 4(c). On average, PIY can perform 29.4% and 14.84% faster than Hadoop-Hash and SkewTune, respectively. Specially, when the data skew is set to 1.2, the improvement is up to 50.65% and 24.54%, respectively. These values demonstrate that the degree of improvements PIY makes is more obvious in heterogeneous cluster than that in homogeneous cluster because it considers the computing capacity of every node during partitioning.

5.5 Grep Benchmark Testing

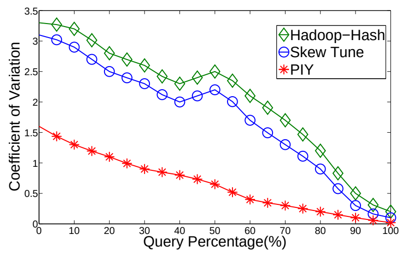
To evaluate the performance of PIY when it deals with the reduce-input-light applications, we run Grep, which is a light job for reduce, in heterogeneous cluster. We improve the Grep benchmark in Hadoop so that it outputs the matched lines in a descending order based on how frequently the searched expression occurs. The data set we use



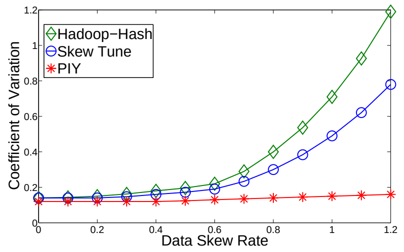
(a)COV of Running Sort in Homogeneous Cluster



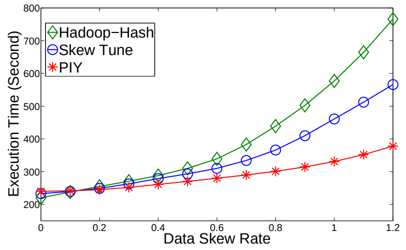
(c)Duration of Running Sort in Homogenous Cluster



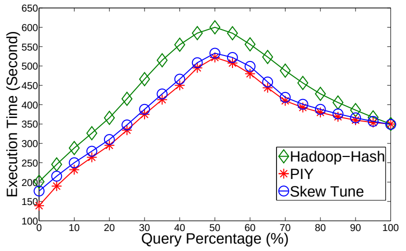
e)COV of Running Grep in Homogeneous Cluster



(b)COV of Running Sort in Heterogeneous Cluster



(d)Duration of Running Sort in Heterogeneous Cluster



(f)Duration of Running Grep in Heterogeneous Cluster

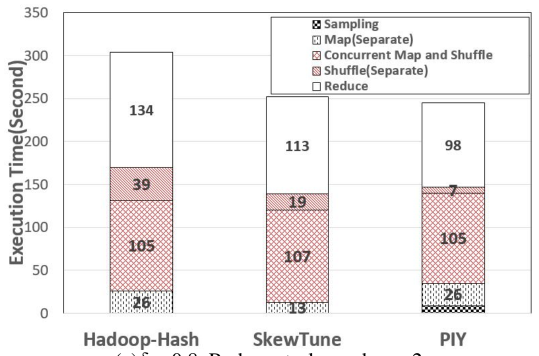
Figure 4: Evaluation of PIY running benchmarks In Hadoop Clusters

is the full English Wikipedia archive with a total data size of 10 GB. Because the behaviour of Grep depends on how frequently the search expression appears in the input files, we tune the expression and make the input query percentages vary from 10% to 100%. Figure 4(e) and (f) show that PIY gets the best performance of COV and job execution time at all time due to the accuracy of PRS and the consideration of heterogeneity. Specially, in Figure 4(e), PIY gets the best COV when the query percentage is lower. This is because the PRS in PIY is good at searching unpopular words in the archive and generates better sampling results. As the query percentage increases, the distribution of the result data becomes increasingly uniform, so the performance gap rapidly closes.

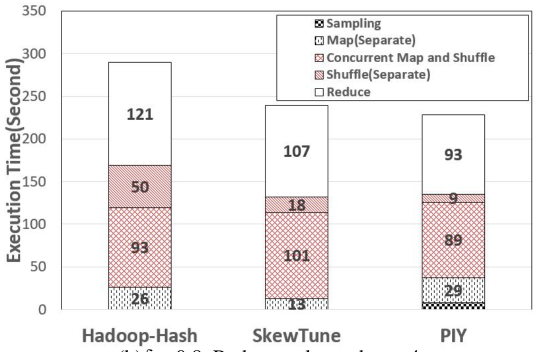
5.6 Optimization In Shuffle Phase

To verify that the BASH algorithm used by PIY can decrease the amount of data transmission in shuffle phase, we record the execution of each phase in MapReduce of Sort job in the heterogenous cluster. Without losing generality, we illustrate the duration time when δ is 0.8 in Figure 5(a). The native Hadoop starts the shuffle tasks when 5% map tasks finish, therefore, we divided MapReduce into 4 phases, which are represented as Map(Seperate), Concurrent Map and Shuffle, Shuffle(Separate), and Reduce. Concurrent Map and Shuffle denotes the overlap period in which the shuffle tasks begin to run and map tasks have not totally finished. Therefore, the duration of Map phase equals the sum of Map(Separate) and 'Concurrent Map and Shuffle'. Similarly, the duration of Shuffle phase, whose fill patterns are red in Figure 5, equals to the sum of Shuffle(Separate) and "Concurrent Map and Shuffle". Specially, because PIY executes PRS in Map phase, its duration of Map phase should contain additional time costed by PRS, which is represented as Sampling in Figure 5. From Figure 5(a), we can see the duration in shuffle phase of PIY is 105+7=112 seconds, which is less than the SkewTune (107+19=126) and Hadoop-Hash (105+39=144), the improvement are $\frac{126-112}{126} = 11.11\%$ and 22.22%.

Through plenty of experiments, we can see that compared with Hadoop-Hash and SkewTune, the improvement degree PIY achieves in shuffle phase is in proportion to the number of reducers until the degree reaches the peak value. In our experiment, the peak improvement degree is achieved when each node can run at 6 map tasks and 4 reduce tasks concurrently. Compared with the original configuration (6 map tasks and 2 reduce tasks), this modification should increase the number of Reducers because the native Hadoop determines which nodes are Reducers according to the computing resource (container in Yarn) in each reducer. The results are shown in Figure 5(b). We can easily find the duration in shuffle phase of PIY is 89+9=98 seconds, which is much less than 119 seconds for SkewTune, 143 seconds for Hadoop-Hash, the improvement is up to $\frac{119-98}{119} = 17.65\%$ and 31.47%, which are larger than Figure 5(a). This can be explained as with the number of Reducers increases, the BASH algorithm finds much input data whose map and reduce tasks are able to be scheduled to the same DataNode. This results in decreasing the amount of data transmission in shuffle phase. However, when we configure each node to run at most 6 map tasks and 6 reduce tasks concurrently, compared with SkewTune and Hadoop-Hash, the improvement caused by PIY are reduced to 12.35% and 19.11%,



(a)δ = 0.8, Reducer task number = 2



(b)δ = 0.8, Reduce task number = 4

Figure 5: The Execution Time Of Each Phase

respectively. How to find the optimal Reducer number is a problem about the tradeoff between the computing parallelism degree and the network transmission amount, and this is our future works on PIY.

5.7 PIY in parallel image processing

Table 2: the size of sample images

image name	size(in bytes)
CARTOSAT-1	1342552576
CARTOSAT-2A	4259355002
CARTOSAT-2B	9204661322

With the need of processing large-sized images increases rapidly, parallel image processing technologies, such as MapReduce, are widely used to shorten the execution time. In this section, we present the experiments conducted for images with large sizes approximately from 1.3 Gigabytes to 9.1 Gigabytes the Chinese Remote Sensing (IRS) satellite series. The sample data sets are shown in Table 2.

We conduct histogram [7] operation on native Hadoop, HIPI [14], which is a open-source Hadoop Image Processing Interface, and PIY. Compared with native Hadoop, HIPI processes image without requiring the additional coding because it implements Java Image Processing Library. Histogram operation counts the frequency of the pixel intensity in an entire image, which is similar to counting the words in the file. In our implementation, the map function splits the large-sized image into several pieces. One piece is processed by one map task, which collects the count of the pixel (gray) value. Reduce function completes the aggregation of the collected numbers from the map functions. To increase the amount of input data in reduce phase, we add TeraSort operation in histogram and finally output the pixel intensity result in descending order. We implement histogram operation in a 5-node heterogeneous cluster, which is composed of the three types of physical computers described in Section 5.1. Specifically, one first type node acts as master, 2 nodes for each of other two types act as slaves.

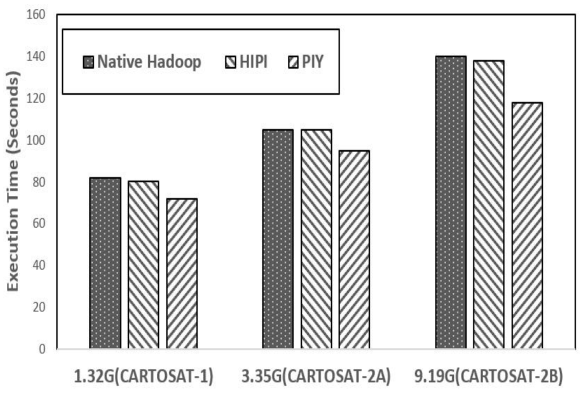


Figure 6: Execution Time Of Histogram

As is shown in Figure 6, PIY gets the shortest execution time when processing all 3 different large-sized images. The execution time is reduced by 11.2%. The reasons are described as follows. First, the distribution of the frequency of the pixel intensity in an large-sized image is not even in general, i.e., the pixel intensity values are skew. Different with native Hadoop and HIPI, PIY considers the data skew by balancing the loads on Reducers. Second, the PRS sampling method helps PIY to realize more accurate distribution of the pixel intensity than the other two frameworks. Third, heterogeneity consideration helps PIY achieve the fastest process speed.

6 Conclusion

This paper proposes PIY to mitigate data skew in MapReduce system. Using the parallel reservoir sampling method we proposed, PIY achieves the distribution of intermediate data accurately with negligible overhead. PIY tries to reduce the network traffic in shuffle phase by decreasing the transmission amount of data on those nodes acting as both Mapper and Reducer. PIY also considers the heterogeneity of the computing resource when balancing load among Reducers. Performance evaluation in both synthetic and real workloads demonstrates that the resulting performance improvement is significant. Compared with some other popular strategies, the improvement PIY achieved reaches 35.62% and 50.65% in homogeneous and heterogeneous clusters, respectively. PIY can also be used in parallel imagine processing to reduce the execution time.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China(No.61272353, No.61428201) and China Postdoctoral Science Foundation(2016M 600912), Program for New Century Excellent Talents in University (NCET-13-0659), Beijing Higher Education Young Elite Teacher Project(YETP0583).

References

[1] L. A. Adamic and B. A. Huberman. Zipfs law and the internet. *Glottometrics*, 3(1):143–150, 2002.

[2] Q. Chen, J. Yao, and Z. Xiao. Libra: Lightweight data skew mitigation in mapreduce. *IEEE Transactions on parallel and distributed systems*, 26(9):2520–2533, 2015.

[3] B. Gufler, N. Augsten, A. Reiser, and A. Kemper. Load balancing in mapreduce based on scalable cardinality estimates. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 522–533. IEEE, 2012.

[4] H. H. Hong Zhang and L. Wang. MRapid: An efficient short job optimizer on hadoop. In *the 31st IEEE International Parallel Distributed Processing Symposium (IPDPS)*. IEEE, 2017.

[5] H. Huang, L. Wang, B. C. Tak, L. Wang, and C. Tang. Cap3: A cloud auto-provisioning framework for parallel processing using on-demand and spot instances. In *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, pages 228–235. IEEE, 2013.

[6] S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi. Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 17–24. IEEE, 2010.

[7] J. N. Kapur, P. K. Sahoo, and A. K. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer vision, graphics, and image processing*, 29(3):273–285, 1985.

[8] C. Kumar. Approximation algorithm project. *arXiv preprint arXiv:1401.2393*, 2014.

[9] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. Skew-resistant parallel processing of feature-extracting scientific user-defined functions. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 75–86. ACM, 2010.

[10] Y. Kwon, M. Balazinska, B. Howe, and J. Rolia. Skewtune: mitigating skew in mapreduce applications. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 25–36. ACM, 2012.

[11] J. Liu, F. Liu, and N. Ansari. Monitoring and analyzing big traffic data of a large-scale cellular network with hadoop. *IEEE Network*, 28(4):32–39, 2014.

[12] V. Subramanian, H. Ma, L. Wang, E.-J. Lee, and P. Chen. Rapid 3d seismic source inversion using windows azure and amazon ec2. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 602–606. IEEE, 2011.

[13] V. Subramanian, L. Wang, E.-J. Lee, and P. Chen. Rapid processing of synthetic seismograms using windows azure cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 193–200. IEEE, 2010.

[14] C. Sweeney, L. Liu, S. Arietta, and J. Lawrence. Hipi: a hadoop image processing interface for image-based mapreduce tasks. *Chris. University of Virginia*, 2011.

[15] J. S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[16] L. Wang, S. Lu, X. Fei, A. Chebotko, H. V. Bryant, and J. L. Ram. Atomicity and provenance support for pipelined scientific workflows. *Future Generation Computer Systems*, 25(5):568–576, 2009.

[17] Z. Xiao, W. Song, and Q. Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE transactions on parallel and distributed systems*, 24(6):1107–1117, 2013.

[18] Y. Xu, W. Qu, Z. Li, Z. Liu, C. Ji, Y. Li, and H. Li. Balancing reducer workload for skewed data using sampling-based partitioning. *Computers & Electrical Engineering*, 40(2):675–687, 2014.

[19] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica. Improving mapreduce performance in heterogeneous environments. In *Osdi*, volume 8, page 7, 2008.

[20] H. Zhang, L. Wang, and H. Huang. Smarth: Enabling multi-pipeline data transfer in hdfs. In *Parallel Processing (ICPP), 2014 43rd International Conference on*, pages 30–39. IEEE, 2014.

Interactive Visualization of <u>R</u>obustness <u>E</u>nhancement in Scale-free <u>N</u>etworks with Limited <u>E</u>dge <u>A</u>ddition (RENEA)

Armita Abedijaberi[1], Nathan Eloe[2], and Jennifer Leopold[1]

[1]Department of Computer Science, Missouri University of Science and Technology, Rolla, MO USA
Email: {*aan87, leopoldj*}*@mst.com*
[2]School of Computer Science and Information Systems, Northwest Missouri State University, Maryville, MO USA
Email: *nathane@nwmissouri.edu*

Abstract

Error tolerance and attack vulnerability of scale-free networks are usually used to evaluate the robustness of these networks. While new forms of attacks are developed everyday to compromise infrastructures, service providers are expected to develop strategies to mitigate the risk of extreme failures. Recently, much work has been devoted to design networks with optimal robustness, whereas little attention has been paid to improve the robustness of existing ones. Herein we present RENEA, a method to improve the robustness of a scale-free network by adding a limited number of edges. While adding an edge to a network is an expensive task, our system, during each iteration, allows the user to select the best option based on the cost, amongst all proposed ones. The edge-addition interactions are performed through a visual user interface while the algorithm is running. RENEA is designed based on the evolution of the network's largest component during a sequence of targeted attacks. Through experiments on synthetic and real-life data sets, we conclude that applying RENEA on a scale-free network while interacting with the user can significantly improve its attack survivability at the lowest cost.

1 Introduction

One of the most important features of large networks is their degree distribution, $P(k)$, or the probability that an arbitrary node is connected to exactly k other nodes. Many real-life networks display a power-law degree distribution with heavy-tailed statistics, which are called scale-free networks. In a scale-free network the probability that a node has k links follows $P(k) \sim k^{(-\lambda)}$, where λ is called the degree exponent and its value is typically in the range between $2 < \lambda < 3$ [1]. Scale-free networks are created by preferential attachment [2], which means newly introduced nodes prefer to connect to existing high-degree nodes. Starting with a small number of nodes, when a new node is added to the network, considering the preferential linking, it will connect to other nodes with the probability proportional to their degree. Coupled with the expanding nature of many networks this explains the occurrence of hubs, which hold a much higher number of links than most of the nodes in the network. Scale-free topology is widely observed in many communication and transportation systems, such as the Internet, World Wide Web, airline networks, wireless sensor networks, and power supply networks, all of which are essential to modern society. One of the most important properties in scale-free networks is the fact that while these networks are strongly tolerant against random failures, they are fragile under intentional attacks on the hubs. Intuition tells us that disabling a substantial number of nodes/edges will result in an inevitable functional disintegration of a network by breaking the network into tiny, non-communicating islands of nodes. However, scale-free networks can be amazingly resilient against accidental failures; even if 80% of randomly selected nodes fail, the remaining 20% still form a compact cluster with a path connecting any two nodes [2]. In fact, the fragileness of scale-free networks

DOI reference number: 10.18293/DMSVLSS2017-015

under intentional attack comes from their heavy-tailed property, causing loss of a large number of links when a hub node is crashed. Hence, the heavy loss of network links quickly makes the network sparsely connected and subsequently fragmented. However, random failures affect mainly the numerous small degree nodes, the absence of which doesn't disrupt the network's integrity.

Considering error tolerance and attack vulnerability, which are two common and important properties of scale-free networks, extensive research efforts have been made to study the robustness of such networks which is defined as the ability of the surviving nodes to remain, as much as possible, interconnected. On that account it is important to understand how to design networks which are optimally robust against malicious attacks, with examples of terrorist attacks on physical networks or attacks by hackers on computer networks. However, it is not possible to abandon the existing networks, which are the result of years of evolution, and rebuild them from the beginning. Hence, it is significantly important to study the optimizing guidelines to enhance the robustness of existing networks in an interactive environment and incorporate the user's input in order to acquire the optimal result at a lower cost.

In this paper, we study the problem of how to improve the robustness of an existing scale-free network and show that its attack survivability can be significantly improved by adding a limited number of edges to it at the lowest cost. This process is carried out by visualizing the process and interacting with the user. There will be no impact on the error tolerance, keeping the degree distribution of the network as much as possible intact.

The organization of this paper is as follows; Section II provides an overview of related work about robustness enhancement in scale-free networks. In Section III we discuss RENEA and its graphical user interface. An example of running RENEA is presented in Section IV. Section V focuses on experiments and results. Finally, conclusions and our plans for future work are presented in Section VI.

2 Related Work

The main approaches to improve robustness of scale-free networks, through topology reconfiguration, can be generally classified into two main categories. The first category involves rewiring edges of the network and the second category suggests addition of edges to the network. Both approaches are carried out in order to obtain a network structure with better robustness. In this section we summarize existing works regarding these two approaches as well as network visualization tools.

2.1 Reconfiguration with Edge Rewiring

In the method proposed in [3], edges are selected randomly to be removed from the graph or to be added to the graph. However, this reconfiguration will change the degree distribution of the graph as well as its diameter, which is not desirable for the network. Another edge rewiring method is proposed in [4], in which two edges A-B and C-D are selected. If A-C and B-D do not already exist in the network, the rewiring operation replaces A-B and C-D with A-C and B-D as long as such reconfiguration does not generate a loop. This rewiring operation obviously does not change any nodal degree since it converts a random network into another one with the same degree distribution. However, for real networks with economic constraints, the nodal degree conservation is not enough due to the cost. In fact, the total length of links cannot be exceedingly large and the number of changes in the network should remain small. In order to minimize the cost, any swap is accepted, only if the increase of the robustness is greater than or equal to a threshold. This procedure is repeated with another randomly chosen pair of edges until no further substantial improvement is achieved. This method results in a network with an onion-like structure [5].

Even though swapping the edges can increase the network robustness, there are some spatially limited real-life networks where edges are hard to re-configure, e.g. power grid networks and the Internet router network. On the other hand, however, there exist the spatial unlimited networks such as airline networks and the Internet switcher network. For spatially unlimited networks whose edges can be easily re-linked, in the Switch Link (SL) method [6], the top P_c fraction of the large-degree nodes are defined as hubs. For each hub, the SL finds two non-hub nodes connecting it. The edge connected to the first non-hub node is kept and the edge connecting the second non-hub node to the first non-hub node is switched. This process will be repeated until all the links connected to the hub nodes are addressed. For the spatially limited networks, the SL method is not economic and feasible since nodes are usually far from each other. In this case, the split hub (SH) method is proposed [6]. This method also starts with defining the top P_c fraction of nodes as hubs and replaces them by a 3-clique which is a complete graph in which every two distinct nodes are adjacent. Then it connects the non-hub nodes, that were connected to the original hub node, to the nodes of the clique randomly.

Edge reconfiguration can change community structure of a network [7]. The community structure refers to the functional modules in the network that play an important role in regards to cascading failures. A network with a strong community structure has few edges between its communities. Hence, its structure is more fragile in terms of attacks on those edges in comparison with networks with a weak community structure. A method [7] is proposed to improve the robustness of a network while preserving its community structure. In this 3-step method, the importance of nodes is calculated based on their degree. Step 1 reconfigures each community to have an onion-like structure [5]. Step 2 swaps edges in such a way that important nodes only connect to the nodes within the same community. Step 3 swaps edges to increase the number of edges between communities. These 3 steps are recursively applied on the network until its robustness hardly increases.

2.2 Reconfiguration with Edge Addition

The number of possible ways of adding an edge to a graph with N nodes and L edges is equal to $\binom{N}{2} - L$. For large real-life (sparse) networks, it is almost infeasible to compare all these possibilities and find the optimal edges to add. Some techniques have been introduced in literature to add edges to an existing graph based on different criteria. Three different enforcing strategies are practiced in [8]. The first method randomly selects a pair of nodes in the network and establishes a new edge between them. The second method selects a pair of nodes with the lowest degree in the network and establishes a new edge between them. Finally, the last method selects a pair of nodes with the highest degree in the network and establishes a new edge between them. According to experiments, the method that prefers low-degree nodes as candidates for adding edges reinforces the attack survivability of the network with a lower cost in comparison with the other methods.

It has been suggested [9] that assortative networks (i.e., high-degree nodes in the network that are more likely linked with other high-degree nodes), are more robust than their disassortative counterparts (i.e., high-degree nodes in the network that are more likely linked with low-degree nodes). Thereupon, a method is proposed [9] to enhance robustness of a network by increasing its assortativity. In this method, a layer index is assigned to each node based on the degree of that node. Accordingly, the layer index for nodes with the lowest degree is 0, for nodes with the second lowest degree is 1, and so on. Then, the probability of adding an edge between a random pair of nodes depends on their layer index difference (i.e., nodes within the same layer are connected with greater probabilities than nodes in different layers); this leads to higher assortativity in the network.

The way that nodes are arranged in a graph is considered as a key factor in overall robustness and efficiency of that graph [10]. The star structure is efficient (the average shortest path length is small), yet fragile in case of removal of the central node. On the contrary, the circle structure is robust with regard to removal of any single node, yet inefficient (the average shortest path length is large). The Node-protecting Cycle method [11] is proposed to combine the properties of both circle and star structures to improve robustness and efficiency of a network.

All of the aforementioned edge rewiring and edge addition algorithms are proposed to improve the robustness of an existing network. However, these works consider one aspect in a network and neglect the others. When it comes to a scale-free network, it is important to take every structural aspect of the network into consideration. In addition, feasibility of a proposed re-configuration of a network must be considered. Having a method to improve the robustness of a scale-free network against malicious attacks while keeping its resilience in case of random failures without disturbing its small-world property at a very low cost is still an unsolved problem. In a scale-free network with the small-world property, the distance between any pairs of nodes is relatively small [12].

2.3 Network Visualization

A number of software tools have been developed to model, analyze, and visualize data that can be represented as a graph or a network. Typically, these tools allow the user to annotate nodes and edges with metadata, and provide utilities such as random graph generation, calculation of analytical measures (e.g., centrality, network distances, PageRank, etc.), and filtering (i.e., viewing only a portion of the graph based on some criteria). Several of these viewers were designed for a particular problem domain, such as finding motifs in gene regulatory networks (e.g., GeneNetWeaver [13]). KDnuggets [14] provides a list of what it considers to be the top 30 network visualization tools that can be used for a wide variety of network applications (e.g., in domains such as biology, finance, and sociology). It is notable that many of these tools are open-source and can be customized to provide additional functionality. The software presented herein differs from other available network visualization tools in the analysis that it facilitates. To the best of our knowledge, there are no other visual network analysis tools available for the purpose of reinforcing robustness of a scale-free network in an interactive environment.

3 Robustness Enhancement Algorithm

The problem setting we consider in this work is to modify a given graph's structure under a given budget to improve its robustness. The budget is often defined in terms of the maximum number of edges that can be added to the network. In practice, the cost of establishing an edge between a pair of nodes is not zero. For real networks, with economical constraints, it is preferable to reduce the overall cost of adding extra edges, keeping the total length (geographically), as low as possible while still achieving the same amount of robustness enhancement. The measure of robustness employed in our algorithm and the edge-addition strategy along with its graphical user interface are specified in the following sections.

3.1 Robustness Metric

The definition of network robustness might change according to a specific application. In this work, the removal of a node from a network is called a “node-knockout” or a “node-attack”, and the robustness of a network is measured by the size of the Largest Connected Component (LCC) in the network after a node-attack [5]. To quantify this, we proceed with a series of node-attacks and subsequently measure the robustness after each node-removal. Hence, the robustness R is defined as:

$$R = \frac{1}{N} \sum_{Q=1}^{N} S(Q)$$

where N is the number of nodes in the network and $S(Q)$ is the fraction of nodes in the LCC after removing Q nodes. The normalization factor $1/N$, makes robustness of networks with different sizes comparable. The minimum value of R is equal to $1/N$, where the network is a star graph and the maximum value of R is equal to 0.5, where the network is a complete graph. A robust network corresponds to a large R value. This distinctive measure is not only simple, but also practical, due to the calculation of the size of the largest component during all possible system-wide failures or intentional attacks.

In our targeted attack scenario, we implicitly admit that the attacker perfectly knows the network's degree sequence and thus can cause maximum damage. An intentional attack is targeted to disrupt the network by removing the most important nodes; here, we find the most connected node, remove it along with all the edges incident with it, calculate $S(Q)$, update the degree sequence for the remaining nodes, and find the new most connected node to repeat the process until the network completely collapses. In case of two or more nodes having the same degree, we simply pick one of them randomly.

3.2 RENEA

Here we present an algorithm called RENEA to improve the robustness of a scale-free network by adding a limited number of edges to it, such that the optimized network, compared to the initial one, has a significantly higher value of robustness. We assume that the ‘defender’ knows about the intention of the ‘attacker’ to cause maximum damage to the network.

For scale-free graphs there is always a very large component which is a connected subgraph that contains a constant fraction of the entire graph's nodes. Resultantly, one can randomly remove more than 80% of the nodes in the graph without destroying that component. Hence, the network will still possess large-scale connectivity [15]. On the other hand, an attack that simultaneously eliminates as few as 10–20$\%$ of the hubs can cause that component to disappear suddenly and break the network into several isolated components. The main idea in our method is to increase the robustness of a network by adding a limited number of edges to it, and therefore to increase the lifetime of the largest component of the graph upon removing high-degree nodes.

Depending upon the nature of a network, adding an edge can be very costly and some networks can only afford a limited number of them. Moreover, in order to keep other structural properties of networks such as their degree distribution as much intact as possible, it requires addition of a limited number of edges to the network. Given that, a threshold parameter is needed for the algorithm to constrain the maximum number of edges that can be added to the network. Furthermore, our iterative algorithm provides the user with a number of edge-additional candidates during each iteration and lets them choose the one that causes the minimum cost in order to be added to the network.

The inputs to our iterative algorithm RENEA are an undirected, unweighted scale-free network represented as a graph $G(V, E)$ with $|V| = N$ nodes and $|E| = L$ edges, the budget δ (maximum number of edges that can be added to G), and θ (the initial percentage of hubs to remove in the attack simulation process). The output from RENEA is a more robust graph $G(V, E')$ with the same number of nodes yet more edges ($L \leq |E'| \leq L + \delta$). The steps of the algorithm are as follows:

```
0) $m = 0$  // m is the number of edges added to G
1) Simulate a targeted attack and remove $\theta$ percent of hubs from $G$.
2) $listComps$ = Sorted list of disconnected components based on their size, in non-increasing order.
3) $comp1$ = The largest component in $listComps$.
4) $comp2$ = The second largest component in $listComps$.
5) Select a node $x \in comp1$
6) Select a node $y \in comp2$
7) $E' = \{x–y\} \cup E$.  // add edge x–y to G
8) $comp = comp1 + comp2$
9) Remove $comp1$ and $comp2$ from $listComps$.
10) Add $comp$ to the beginning of $listComps$.
11) $m = m + 1$
12) Repeat steps 3-11 until $|listComps| == 1$ or $m == \delta$.
```

RENEA starts off with simulating an attack on graph G (line 1). There are two common types of attacks that can be carried out on a network known as serial-attack and sudden-attack. In both of these attacks an initial θ percent of highest-degree nodes are to be removed. In the sudden-attack, θ percent of highest-degree hubs are identified and removed at once, whereas in the serial-attack, which is known to be more damaging, hub removal happens a bit differently. In the serial-attack, first the highest-degree hub is removed, then the degree of each remaining node is recalculated and among them again the highest-degree hub is removed until θ percent of hubs are discarded. In the graphical user interface, a user can choose either of these options to simulate an attack on the network.

Once G is fragmented, all disconnected components are found, where the number of nodes in each component determines the size of that component. In order to add the least number of edges yet gain the highest improvement in robustness, the single-node components will be ignored. $listComps$ contains a list of components sorted based on their size in a descending order (line 2). The first and second members of $listComps$, the largest and second largest components, are called $comp1$ and $comp2$, respectively (lines 3-4).

The algorithm adds an edge between node x in $comp1$ and node y in $comp2$. There exists a list of candidates whose size is equal to $|comp1| \times |comp2|$. Our user interface allows the user to pick one edge that can impose the least cost to the network (lines 5-7). Afterwards, $comp1$ and $comp2$ are combined into $comp$ and both are removed from $listComps$, and $comp$ is added to the beginning of $listComps$ (to keep it sorted) (lines 8-10). Variable m is used to keep track of the number of edges added to G (line 11). At this point one edge is already added to G ($E' = E \cup \{x\text{–}y\}$) and m=1. RENEA, as an iterative algorithm, repeats steps 3-11 until $listComps$ contains only one component (at the end of each iteration, the size of $listComps$ decreases by one) or the desired number of edges are added to G ($|E'| = L + \delta$) (line 12).

3.3 Edge Removal

Even though applying RENEA on a graph improves its robustness remarkably, adding more edges to a network comes with an extra cost. Hence, depending on the nature of the network, some users may or may not decide to mitigate the total cost by getting rid of some edges from the graph yet still have a considerable overall enhancement in the robustness of the network.

Upon request of the user, our algorithm nominates some edges to get removed from the graph based on the betweenness centrality value of them. The edge betweenness centrality is defined as the number of the shortest paths that pass through an edge of a network. An edge with a high edge betweenness centrality score represents a bridge-like connector between two parts of a network and the removal of such edges may affect the communication between many pairs of nodes through the shortest path between them.

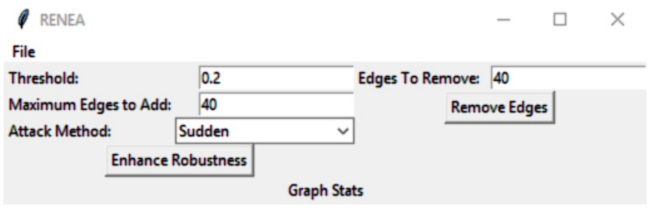
To implement the edge-removal part, first each edge is associated with an edge centrality value. Then these values are sorted in increasing order. The user can request the maximum number of edges that s/he is willing to remove from the network. Removing edges with high betweenness, that occupy critical roles in the network, can force many pairs of nodes to be re-routed on a longer way in order to communicate with each other. It can also degrade the overall efficiency of the network in terms of communication. Hence, the algorithm starts removing edges with the lowest betweenness value, as long as that edge-removal does not make the network disconnected, until the desired number of edges are removed.

3.4 The Graphical User Interface

RENEA is implemented using the Python programming language using the powerful NetworkX [16] library to perform graph operations. The Graphical User Interface (GUI) uses elements of the networx_viewer to easily enable an interactive view of the graph that allows scrolling, zooming, and moving nodes. The Graph Viewer element is included in a Tkinter based GUI; Tkinter is the standard GUI library in Python and is included in a variety of distributions of the language. This helps to ensure that the application is as cross platform as possible.

The RENEA user interface [*Figure 1*] consists of the following controls: (i) a file chooser to allow the user to select a file that contains a list of nodes and edges of the graph, (ii) a text input field to specify the initial value of the threshold (θ), (iii) a text input field to specify the maximum number of edges to be added to the graph, (vi) a drop-down menu to select the desired type of attack on the graph, (vi) a control button to start the process of adding edges to the graph to enhance robustness, (v) a text input field to specify the maximum number of edges to be removed from the graph, and (vi) a control button to start the process of removing edges from the graph.

As shown in [*Figure 2*], a data set has been selected, and all of its specifications are presented in the GUI such as the number of nodes, edges, and initial robustness. The important nodes in terms of their degree (hubs) are presented using bigger circles. In this GUI the user can drag and relocate each edge and node, and zoom in/out on the graph for a closer look.



Threshold: 0.2
Edges To Remove: 40
Maximum Edges to Add: 40
Attack Method: Sudden

Figure 1: RENEA GUI

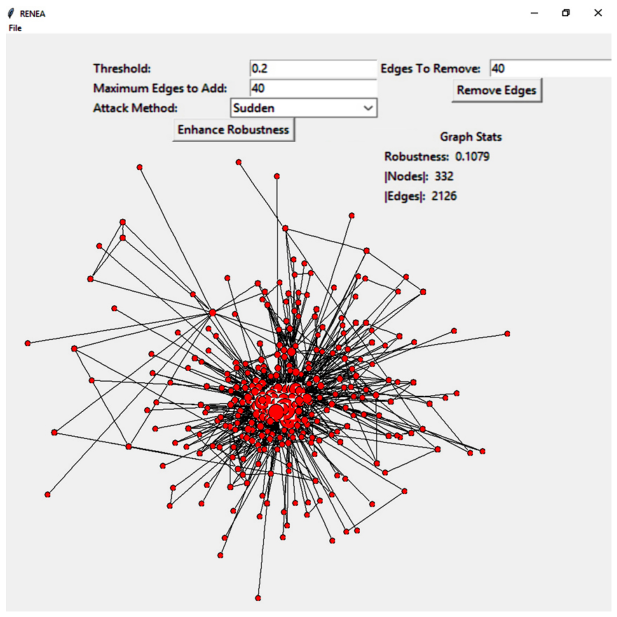
4 A Running Example In RENEA

To demonstrate the basic concepts behind RENEA, here we walk through a simple example. We start by uploading a graph in the GUI with 20 nodes, 21 edges, and the initial robustness of 0.1052. Once the graph is uploaded, all of its specifications are presented in the GUI [*Figure 3*]. For this particular graph, we set *'Threshold' = 20%*, *'Maximum Edges to Add' = 3*, and *'Attack Method' = Serial*. Once the user hits the *'Enhance Robustness'* button, the algorithm runs and suggests a list of edge IDs to be added. The user can accept or discard the proposed edges [*Figure 4*]. Once the user accepts these edges they will be added to graph (they are displayed with thicker lines in blue) [Figure 5]. If the user chooses to delete 3 edges, once they hit the *'Remove Edges'* button, the algorithm calculates the betweenness of the edges and nominates a maximum of three edges with the lowest betweenness whose removal does not disconnect the graph. These edges are displayed in red with thicker lines [*Figure 6*]. The GUI also shows the amount of decrease in robustness due to edge removal. Once the user accepts the changes those edges get removed and the updated robustness along with the final graph is displayed [*Figure 7*].

5 Experiments

5.1 Dataset

In this section, we experimentally examine the performance of RENEA, and for performance evaluation comparison, we have also implemented three other edge-addition algorithms: one to add edges between randomly selected pairs of nodes, one to add edges between nodes with high degrees only, and finally one to add edges between low-degree nodes. For the experiment, we used a real-life American Airlines dataset which is an unweighted, undirected, and connected scale-free graph. This dataset contains 332 airports (nodes) and 2126 flights between airports (edges) and contains no multi-edges (two or more edges that are incident to the same two nodes) or self-loops (an edge from a node to itself).

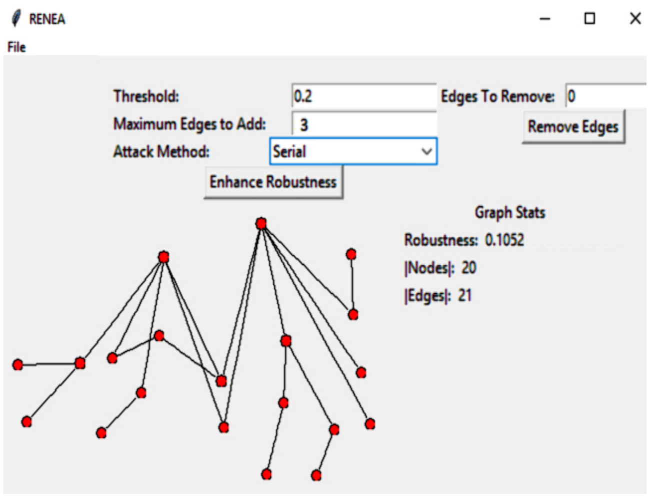


Threshold: 0.2
Edges To Remove: 40
Maximum Edges to Add: 40
Attack Method: Sudden
Robustness: 0.1079
|Nodes|: 332
|Edges|: 2126

Figure 2: RENEA GUI after uploading a dataset containing USAir network; all the specifications of this graph (e.g., number of nodes, number of edges, and initial robustness) also are displayed.

5.2 Results

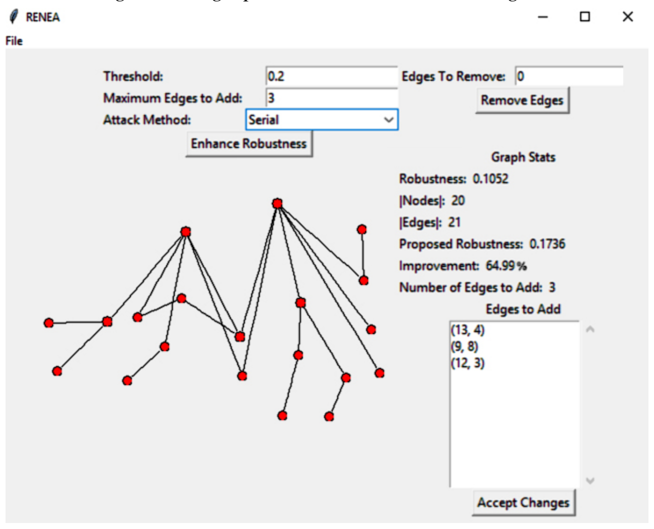
We tested RENEA and three other algorithms on the American Airlines network. We ran each algorithm 10 times and took the average of improvement acquired by each. For each experiment, we added 20, 30, 40, 50, and 60 edges to the original graph using different algorithms and computed the obtained robustness. As shown in [*Table 1*], RENEA significantly outperforms the other algorithms, accomplishing two times more improvement in comparison with the other methods. Results show that by adding less than 3% of edges to the graph, robustness improves up to 70%, whereas the random edge addition can only improve the robustness up to 35%. [*Figure 8*] shows how the size of the biggest component in the American Airlines network is changing while removing q hubs from the original network. The size of the largest component after applying RENEA causes the graph to hold its integrity for a longer time as opposed to using the other methods. [*Figure 9*] compares the degree distribution of the US Air network before and after adding 50 edges via applying RENEA. Because of adding a limited number of edges to the graph, the shape of its degree distribution remains almost the same. The reason that a scale-free network is robust in terms of random failure is because having a power-low degree distribution and adding a limited number of edges to the graph does not cause severe changes in it. The goal is to keep all the unique properties of the graph the same while enhancing its robustness against targeted attacks.



Threshold: 0.2
Maximum Edges to Add: 3
Attack Method: Serial
Edges To Remove: 0

Graph Stats
Robustness: 0.1052
|Nodes|: 20
|Edges|: 21

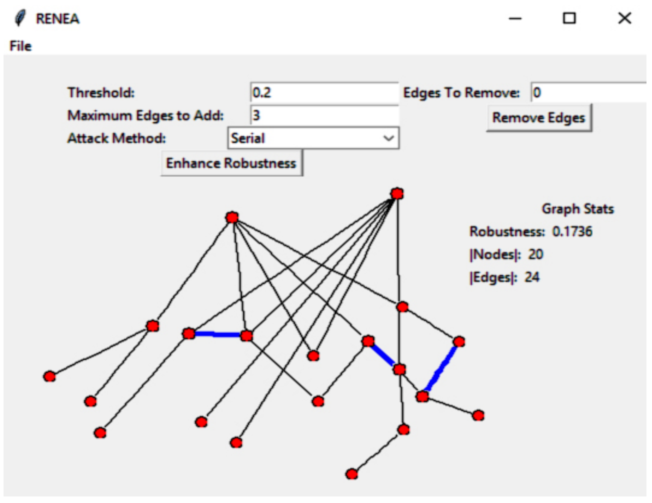
Figure 3: A graph with 20 nodes and 21 edges.



Threshold: 0.2
Maximum Edges to Add: 3
Attack Method: Serial
Edges To Remove: 0

Graph Stats
Robustness: 0.1052
|Nodes|: 20
|Edges|: 21
Proposed Robustness: 0.1736
Improvement: 64.99 %
Number of Edges to Add: 3
Edges to Add
(13, 4)
(9, 8)
(12, 3)

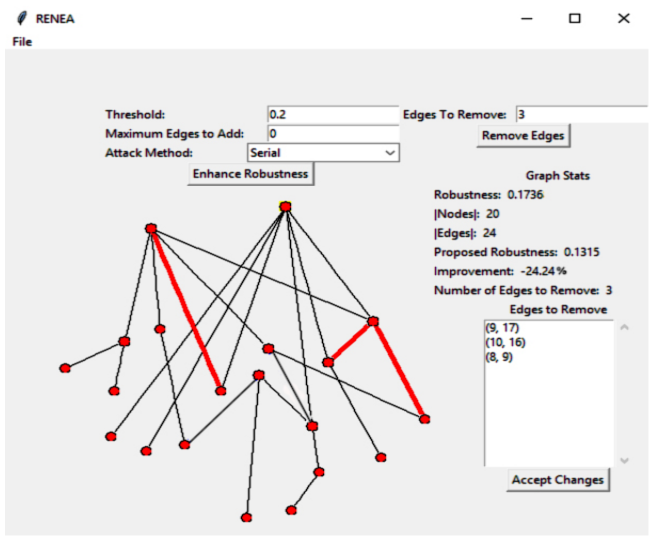
Figure 4: A list of suggested edges to add to enhance robustness.



Threshold: 0.2
Maximum Edges to Add: 3
Attack Method: Serial
Edges To Remove: 0

Graph Stats
Robustness: 0.1736
|Nodes|: 20
|Edges|: 24

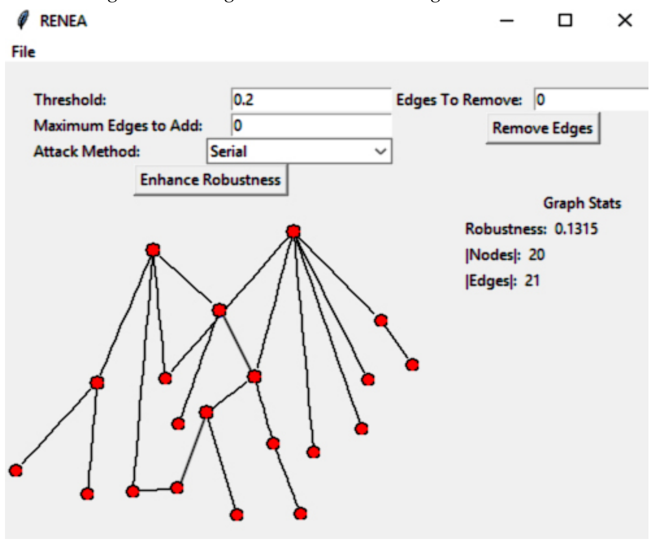
Figure 5: Graph after adding 3 edges.



Threshold: 0.2
Maximum Edges to Add: 0
Attack Method: Serial
Edges To Remove: 3

Graph Stats
Robustness: 0.1736
|Nodes|: 20
|Edges|: 24
Proposed Robustness: 0.1315
Improvement: -24.24 %
Number of Edges to Remove: 3
Edges to Remove
(9, 17)
(10, 16)
(8, 9)

Figure 6: 3 edges are nominated to get removed.



Threshold: 0.2
Maximum Edges to Add: 0
Attack Method: Serial
Edges To Remove: 0

Graph Stats
Robustness: 0.1315
|Nodes|: 20
|Edges|: 21

Figure 7: After adding 3 edges and removing 3 edges, the overall robustness improvement is 25%.

6 Conclusion and Future work

Real-life complex networks are known to be resilient in terms of random failures yet fragile in the terms of targeted attacks. To enhance their total robustness, one strategy is to add more edges to the network and the other is to rewire some qualified edges in the network. Based on the type of the network either of these methods could be used. Mostly these strategies accept a change only if it improves the robustness of the network by a threshold. Hence, it requires them to compute the robustness of the graph before and after applying a change which could be very time-consuming depending on the size of the graph. So they are not considered efficient in terms of time. Additionally,

Table 1: The performance of RENEA vs. three other algorithms Algo1, Algo2, and Algo3. Algo1 adds edges randomly, Algo2 adds edges among low-degree nodes, and Algo3 adds edges among high-degree nodes. The dataset is an American Airlines dataset with 332 nodes, 2126 edges and initial robustness of 0.1079. We added 20, 30, 40, 50, and 60 edges to the original graph and the averages of total robustness improvement acquired after running each algorithm are presented.

Initial R 0.1079	20 edges	30 edges	40 edges	50 edges	60 edges
RENEA	26.17%	38.16%	49.94%	55.25%	66.09%
Algo1	11.08%	18.19%	23.20%	30.44%	34.07%
Algo2	11.55%	13.17%	15.42%	21.80%	24.60%
Algo3	10.26%	12.14%	19.09%	21.51%	21.72%

none of these solutions take the nature of the network into consideration. Thus, their proposed solution is not always a practical one.

In this work, we presented RENEA, an iterative algorithm that is designed to enhance the overall robustness of a scale-free network against malicious attacks by adding a limited number of edges to it. Adding new connections to the nodes arbitrarily and without any constraint can change the nodal degree of the graph and disturb other structural properties of it. We also presented a user interface for RENEA that allows the user to see the effect of the changes to the network. In addition to the excellent performance of RENEA, the fact that it does not compute the robustness during each iteration, makes it work very fast to communicate with the user. During each iteration, RENEA suggests the best edges such that adding them can best increase the robustness of a graph, and the user, considering the cost, can accept or reject them. Finally, if the user also wants to remove some edges from the graph to mitigate the overall cost, RENEA nominates the ones with low betweenness centrality value whose removal does not make the graph disconnected. Our future work will be focused on performing formal studies of the usefulness and usability of the user interface as well as further improvement in the performance of RENEA.

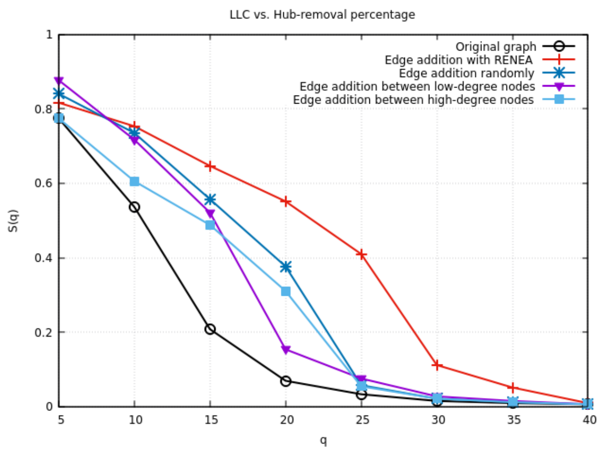


Figure 8: X-axis represents the number of removed nodes and Y-axis represents the size of LLC in the American Airlines network before and after reconfiguration. The line in red shows the performance of RENEA.

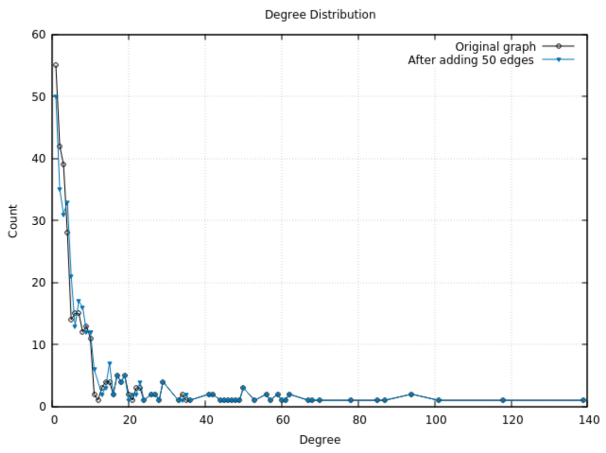


Figure 9: X-axis represents degrees of nodes and Y-axis represents how many nodes share the same degree. After adding 50 edges to the US Air network, its degree distribution remains almost the same.

References

[1] Cohen, Reuven, and Shlomo Havlin. "Scale-free networks are ultrasmall." Physical review letters 90.5 (2003): 058701.

[2] Barabsi, Albert-Lszl, and Rka Albert. "Emergence of scaling in random networks." science 286.5439 (1999): 509-512.

[3] Beygelzimer, Alina, et al. "Improving network robustness by edge modification." Physica A: Statistical Mechanics and its Applications 357.3 (2005): 593-612.

[4] Xiao, S., et al. "Robustness of scale-free networks under rewiring operations." EPL (Europhysics Letters) 89.3 (2010): 38002.

[5] Schneider, Christian M., et al. "Mitigation of malicious attacks on networks." Proceedings of the National Academy of Sciences 108.10 (2011): 3838-3841.

[6] Ze-Hui, Qu, et al. "Enhancement of scale-free network attack tolerance." Chinese Physics B 19.11 (2010): 110504.

[7]] Yang, Yang, et al. ”Improving the robustness of complex networks with preserving community structure.” PloS one 10.2 (2015): e0116551.

[8] Zhao, Jichang, and Ke Xu. ”Enhancing the robustness of scale-free networks.” Journal of Physics A: Mathematical and Theoretical 42.19 (2009): 195003.

[9] Wu, Zhi-Xi, and Petter Holme. ”Onion structure and network robustness.” Physical Review E 84.2 (2011): 026106.

[10] Chan, Hau, Shuchu Han, and Leman Akoglu. ”Where graph topology matters: the robust subgraph problem.” Proceedings of the 2015 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2015.

[11] Li, Li, et al. ”Enhancing the Robustness and Efficiency of Scale-free Network with Limited Link Addition.” TIIS 6.5 (2012): 1333-1353.

[12] Wang, Xiao Fan, and Guanrong Chen. ”Complex networks: small-world, scale-free and beyond.” IEEE circuits and systems magazine 3.1 (2003): 6-20.

[13] Schaffter, Thomas, Daniel Marbach, and Dario Floreano. ”GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods.” Bioinformatics 27.16 (2011): 2263-2270.

[14] KDNuggets. Top 30 Social Network Analysis and Visualization Tools, June 2015. http://www.kdnuggets.com/2015/06/top-30-social-network-analysis-visualization-tools.html. Accessed March 28, 2017.

[15] Callaway, Duncan S., et al. ”Network robustness and fragility: Percolation on random graphs.” Physical review letters 85.25 (2000): 5468.

[16] Schult, Daniel A., and P. Swart. ”Exploring network structure, dynamics, and function using NetworkX.” Proceedings of the 7th Python in Science Conferences (SciPy 2008). Vol. 2008. 2008.

Effective Path Summary Visualization on Attributed Graphs

Duncan Yung and Shi-Kuo Chang
Department of Computer Science
University of Pittsburgh, PA, USA
{duncanyung,chang}@cs.pitt.edu

ABSTRACT

When a new dataset is modeled as an attributed graph or users are not familiar with the data, users may not know what can be retrieved from the attributed graph. Sometimes, users may have some intuition about the query, but how to exactly formulate queries (e.g. what attribute constraints to use) is still unclear to users. In this paper, we propose the idea of attributed path summary. In general, attributed path summary is a grouping of vertices such that vertices in each group contain paths from source to destination and the entropy of attributed values within a group is low and biased toward the intuition (i.e. preferred attribute values) given by users. We propose a novel 3-phrase approach which stitches key vertices together to form candidate paths and inflates those candidate paths into path summary. An extensive case study and experimental evaluation using the real Facebook graph that visualizes the path summary demonstrates the usefulness of our proposed attributed path summary as well as the superiority of our proposed techniques.

1. INTRODUCTION

Attributed graph is widely used for modeling a variety of information networks [11, 15], such as the web, sensor networks, biological networks, economic graphs, and social networks. When a new dataset is modeled as an attributed graph or users are not familiar with the data, users may not know what can be retrieved from the attributed graph. Sometimes, users may have some intuition about the query, but how to exactly formulate queries (e.g. what attribute constraints to use) is still unclear to users.

In this paper, we propose the idea of visualizable attributed path summary. In general, an attributed path summary is a grouping of vertices such that vertices in each group contain a path from source to destination and the entropy of attributed values within a group is low and biased toward the intuition (i.e. attribute values) given by users. In addition, we argue that a visualizable attributed path summary can be easily visualized and understood by users.

1.1 Application Scenario

Social Network: For example, an FBI agent has a social network, but he/she is not familiar with the attribute values and graph structure of the social network. The agent wants to investigate the relationship between Duncan and a terrorist leader using the social network as the FBI believes that social network would contain a lot of useful insight for investigation. The agent just got an intuition that people between Duncan and the terrorist leader may live in the country C_1 and believe in religion R_1,R_2. The attributed path summary query computes a summary of paths from Duncan to the terrorist leader that are close to the offered attribute values (i.e. C_1,R_1 or R_2). The path summary offers insight for the agent to formulate different path queries for investigation.

Metabolic Network: In metabolic networks, each vertex is a compound, and an edge between two compounds indicates that one compound can be transformed into another one through a certain chemical reaction. Vertex attributes can be features of the compound; edge attributes can be details of a chemical reaction between two compounds. A reachability query on metabolic networks discovers whether compound A can be transformed to compound B under given path attribute constraints. A biologist wants to study how to transform compound A to compound B. The biologist only knows that cost-to-trigger-reaction has to be around \$10. The attributed path summary computes a summary of paths from compound A to compound B that are close to the offered attribute value (e.g. cost-to-trigger-reaction$\approx$ \$10). The path summary offer insight for the biologist to formulate path queries for the study.

1.2 Challenges

Nowadays, a big graph with a few million vertices is common, and that results in an exponential number of paths between any two vertices. A large number of possible paths between 2 vertices makes computing path summary a challenging task.

Among a huge number of possible paths between 2 vertices, which type of path the user prefers is unknown since even the user is not familiar with the graph, and he/she may not know what he/she can get from the graph. Therefore, our task is to compute a path summary for the user.

Computing an effective summary for a user is non-trivial as no user would prefer to read a lot of text to understand the summary. Hence, an effective path summary is a summary that can be easily visualized by users. Visualizing a large portion of the graph is not feasible as that would overwhelm the user. On the contrary, if the summary is too concise, the user may not get the information he/she wants.

10.18293/DMSVLSS2017-019.

1.3 Our Contributions

Our first contribution is to introduce and define the attribute path summary query on attributed graph problem. We define attributed path summary to be groups of vertices that contain users' intuition as well as satisfy some path properties. The users' intuition is expressed as hints for computing the path summary. Users can offer whatever attribute values that they consider as the hint. These summaries offer insight to users about the attribute values and connection between the given source and destination vertices.

Our second contribution is to propose an efficient and effective approach for finding attributed path summary. Our proposed approach consist of three phrases. The first phrase efficiently finds all key vertices that have attribute values belonging to the hint offered by the user. Including key vertices ensures the summary would represent paths with attribute values that are close to the intuition of users. Then, based on those key vertices, a novel stitching algorithm is proposed to connect the source, the destination, and key vertices together to form a relatively small key vertex graph. The stitching algorithm finds paths with a small variation in attribute values between key vertices so that users can easily understand the attribute distribution between key vertices. After that, high-quality candidate paths between the source and the destination are found on that small key vertex graph efficiently. Finally, candidate paths are inflated to vertex groups by greedily including adjacent vertices. Including adjacent vertices would offer more attribute values choices for users to formulate their queries.

1.4 Paper Organization

Section 2 talks about related works. Section 3 presents definitions and problem statement. Section 4 gives details of our approach for finding attributed path summary. Section 5 presents an extensive experimental evaluation for the proposed approach. We conduct case studies on the Facebook graph that visualize our path summary results for illustrating the effectiveness of our proposed path summary. We also conduct experiments to study the change in entropy and execution time under different parameter settings. Finally, Section 6 concludes this paper.

2. RELATED WORK

In this section, we present a summary of related works.

2.1 Attributed Graph Summarization

Graph summarization has been extensively studied [10, 13, 13, 17, 14, 16, 4, 7, 12, 3, 2], and various ways of summarizing graphs have been proposed. Grouping-based summarization methods [10, 13, 13, 17, 14] takes into account both graph structure and attribute distributions for aggregating vertices into supernode and superedges; compression-based summarization methods [16, 4, 7] exploit the MDL principle to guide the grouping of vertices or the discovery of frequent sub-graphs to form a graph summary; influence-based summarization methods [12] leverage both graph structure and vertex attribute value similarities in the problem formualtion so as to summarize the influence process in a network; pattern-mining-based summarization methods [3, 2] identify frequent graph structural patterns for aggreagate into supernodes so as to reduce the size of the input graph and as a result, improving query efficiency. These techniques focus on computing summary for the whole graph. On the other hand, our techniques focus on computing visualizable path summary between two vertices that users are interested in.

2.2 Attributed Graph Clustering

Zhou et al [18] proposed $SACluster$, which is an attributed graph clustering algorithm based on both graph structural and attribute similarities through a unified distance measure. Zhou et al [18] proposed first to partition a large graph associated with attributes into k clusters so that each cluster contains a densely connected subgraph with homogeneous attribute values. Then, an effective method is used to automatically learn the degree of contributions of structural similarity and attribute similarity. Zhou et al [19] further improve the efficiency and scalability of $SACluster$ [18] by proposing an efficient algorithm $IncCluster$ to incrementally update the random walk distances given the edge weight increments.

One fundamental difference between summarization and clustering is that former finds coherent sets of vertices with similar connectivity patterns to the rest of the graphs, while clustering aims at discovering coherent densely-connected groups of vertices [8]. Similar to graph summary, graph clustering only computes a summary of the whole graph while our techniques focus on a summary of paths between two vertices.

2.3 Graph Visualization

The size of the graph to view is a key issue in graph visualization [6]. To deal with this, researchers proposed a lot of techniques in graph drawing [6], such as H-tree layout, radial view, balloon view, tree-map, spanning tree, cone tree, hyperbolic view, as well as methods for reducing visual complexity [9], such as clustering, sampling, filtering, partitioning. We argue that simply applying those graph drawing technique cannot handle big attributed graphs with million of vertices and edges as these methods are too general. For existing visual complexity reduction methods, how to effectively applying them to our problem needs further investigation.

3. PRELIMINARIES

3.1 Problem Statement

DEFINITION 1. **[Attributed Graph]** *An attributed graph [15] G, is an undirected graph denoted as $G = (V, E, A_v)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $A_v = \{A(v)\}$ is a set of d_v vertex-specific attributes, i.e. $\forall v \in V$, there is a multidimensional tuple $A(v)$ denoted as $A(v) = (A_1(v), A_2(v), ..., A_{d_v}(v))$.*

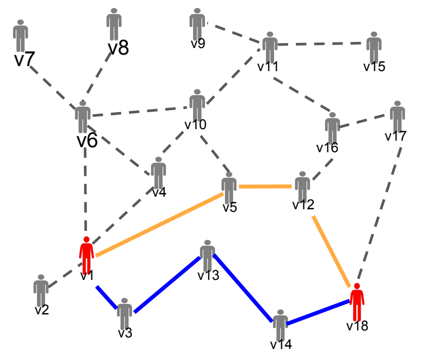
DEFINITION 2. **[Attribute Hint** H**]** *is a set of distinct attribute values.*

$$H = \{H_1, H_2, .., H_{d_v}\}$$

DEFINITION 3. **[Contain Function** $\phi(P_i, H)$**]**

$$\phi(P_i, H) = \sum_{j=1}^{|P_i|} contain(v_j, H)$$

$$contain(v_j, H) = \begin{cases} 1, & \forall k = 1..d_v \;\; if \; \exists A_k(v_j) \in H_k \\ 0, & otherwise \end{cases}$$



vID	Country	Job
$v1$	USA	IT
$v3$	USA	IT
$v5$	USA	IT
$v12$	SG	IT
$v13$	JAP	FIN
$v14$	UK	EDU
$v18$	USA	FIN
..	...	...

Figure 1: Path Summary (P_1-blue, P_2-orange)

For example in Figure 1, given that $H = \{\{USA, SG, JAP\}, \emptyset\}$,$P_1 = \{v1, v3, v13, v14, v18\}$, and $P_2 = \{v1, v5, v12, v18\}$, $\phi(P_1, H) = 1+1+1+0+1 = 4$ and $\phi(P_2, H) = 1+1+1+1 = 4$.

DEFINITION 4. **[Attributed Path Summary $PSum(G, s, t, l, H)$]** *For an attributed graph G, $PSum(G, s, t, l, H)$ is a set of vertices $\{P_1, P_2, ..., P_k\}$ such that:*

1. *$\forall\ v \in P_i$ are connected,*
2. *$\exists\ v, v' \in P_i$, v is adjacent to s and v' is adjacent to t,*
3. *$\forall P_i, P_j \in G, i \neq j, P_i \cap P_j = \emptyset$,*
4. *$\forall P_i, \phi(P_i, H) \geq l$,*
5. *$\forall v \in P_i, dist(s, v) + dist(v, t) \leq l$, and*
6. *$\nexists P \in G \wedge P \notin PSum(G, s, t, l, H)$ satisfing condition 1 to 5.*

where $\phi(P_i, H)$ is the quality of the summary and $dist(v, v')$ is the shortest distance from v to v'.

Continuing the above example, in Figure 1, given that $l = 4$, there are two paths P_1, P_2 in the attributed path summary. They are $P_1 = \{v1, v3, v13, v14, v18\}$ and $P_2 = \{v1, v5, v12, v18\}$. For example, all vertices in P_1 are connected, $\exists\ v_3$ and v_14 adjacent to s and t, $P_1 \bigcap P_2 = \emptyset$, $\phi(P_1, H) = \phi(P_2, H) = 4 = l$, and for all $v_i \in P_1, P_2$, and $dist(s, v_i) + dist(v_i, t) \leq 4$.

Problem Statement

[Attributed Path Summary Query q_p] Given an attributed graph $G = (V, E, A_v)$, source s, destination t, attribute hint H, and lower bound of number of vertices in every P_i that contains at least one attribute value in attribute hint l, q_p return an attributed path summary $PSum(G, s, t, l, H)$.

3.2 Quality of Path Summary

The quality of path summary is defined as the entropy in [18] and is reformulated in definition 5.

DEFINITION 5. **[Path Summary Quality]**

$$entropy(P_j) = \sum_{i=1}^{d_v} entropy(a_i, P_j)$$

where

$$entropy(a_i, V_j) = -\sum_{n=1}^{n_i} p_{ijn} log_2 p_{ijn}$$

and k is the number of P_i in $PSum$, p_{ijn} is the percentage of vertices in P_j which have value a_n on attribute a_i.

For example, $entropy(Country, P_1) = -(\frac{3}{5}log_2\frac{3}{5} + \frac{1}{5}log_2\frac{1}{5} + \frac{1}{5}log_2\frac{1}{5})$.

4. COMPUTING PATH SUMMARY

In this section, we introduce our path stitching approach for computing attributed path summary effectively based on attribute hint.

4.1 Algorithm Design

Our heuristic approach has the following steps and design principles.

1. Firstly, we want to find all vertices - key vertices, that are related to the given attribute hint. The search of key vertices ensures that all vertices that match any attribute value in the hint and fulfill the distance requirement (Condition 5, Definition 4) are used for computing a path summary.

2. Given those key vertices, we perform a concurrency graph traversal that systematically stitches key vertices, the source, and the destination. Using stitched key vertices, we find candidate paths that go from the source to the destination via key vertices based on the entropy of attribute values on the path. Key vertices are vertices that users care and want to see in the visualized path summary. The stitching algorithm can effectively connect key vertices so that attribute values on the path between key vertices are consistent. That offers a clear view for users to understand the attribute distribution between key vertices.

3. Finally, given the candidate paths, we perform a candidate path inflation for computing the path summary. Candidate path inflation includes vertices close to vertices in candidate paths into the candidate paths. That allows users to understand attribute distributions around key vertices. When users are considering what attribute constraint to use for their attribute graph queries, they can consider attribute values on candidate paths as well as attribute values close to the candidate path as an alternative.

Algorithm details are presented in below sections with conceptual examples.

4.2 Finding Key Vertices

We first introduce the concept of key vertex (Definition 6). Then, we present two steps that exploit existing approach to efficiently find all key vertices.

DEFINITION 6. **[Key Vertex v^k]** *is a vertex that has at least one attribute value belonging to an attribute value in the attribute hint H.*

$$\forall i = 1..d_v \forall j = 1..d_v\ \ \exists A_i(v^k) \in H_j$$

$$where\ \ H_j \in H$$

The first step is to retrieve all key vertices. Traditional indexes that support range query (e.g. B+ tree) can be used to index each attribute. Given H, for each non-empty $S_j \in H$, we query the corresponding index for a set of vertices that have attribute values in S_j. Then, we do a union of all these vertices and get the key vertex set V_k. After that, all vertices v that does not satisfy $dist(s, v) + dist(v, t) \leq l$ are filtered out.

For example, in Figure 1, if the hint contains only $Country = USA$, all vertices with attribute value $Country = USA$ (e.g. v_1, v_3, v_5, v_{18}) are retrieved from the precomputed index (e.g. B+ tree).

4.3 Finding Candidate Path

After all key vertices are found, the second step is to find candidate paths that satisfy constraints in Definition 4.

4.3.1 Stitching Algorithm

Since key vertices are essential (so as to satisfy condition 4 in Definition 4) for paths in path summary, we do not want to find candidate paths that do not contain any key vertex. The idea of the stitching algorithm is to connect $s - t$ and key vertices so as to form candidate paths. During the graph traversal, entropy and hop distance values are taken into account.

Algorithm 1 is the pseudo code of the stitching algorithm. The stitching algorithm first puts s, t, and all key vertices (lines 7-13) into the priority queue. Each node in the priority queue contains the current vertex, a key vertex, parent of current vertex, the distance from a key vertex to the current vertex, and the entropy of path from a key vertex to the current vertex, where the entropy value is used to determine the priority.

Then, the graph is traversed starting from each of the key vertices. When the algorithm reaches a visited node $cur.v$ (line 16), if key vertex of current node's parent ($key1$) is not equal to the key vertex of curent node ($key2$) (line 19), the path from $key1$ to $key2$ is recovered and put into $PathMap$ (line 21), edges between vertice $key1$ and $key2$ are added in to $KeyVertexG$ (lines 22-23), and the algorithm continue (line 22); when the algorithm reaches a non-visited node (line 25), parent and key vertex of current node is saved (line2 26-27) and current node becomes visited (line 28).

After that, adjacent neighbors of $cur.v$ that satisfy the upper bound distance constraint (line 30) are put into the priority queue (line 33), where the entropy of the path from the key vertex to $cur.v$ as well as the distance are taken into account. Finally, the graph traversal continues until the priority queue becomes empty.

A conceptual example will be presented in Section 4.3.3.

Algorithm 1 Stitching Algorithm

```
 1: procedure STITCHING($G, V_k, s, t, l$)
 2:     $Array < bool > \ \ visited$
 3:     $Array < int > \ \ parents$
 4:     $Array < Array < int >> \ \ KeyVertexG$
 5:     $Array < int > \ \ keys$
 6:     $priority_queue < node > \ \ qu$          ▷ lower entropy first
 7:     $node \ \ src(s, s, s, 0, 0)$                                  ▷
            $(vert., keyVert, parent, dist, entropy)$
 8:     $qu.push(src)$
 9:     $node \ \ dest(t, t, t, 0, 0)$
10:     $qu.push(dest)$
11:     for all $v^k \in V_k$ do
12:         $node \ \ n(v^k, v^k, v^k, 0, 0)$
13:         $qu.push(n)$
14:     while !$qu.empty()$ do
15:         $cur \leftarrow q.pop()$
16:         if $visited[cur.v] == true$ then
17:             $key1 \leftarrow keys[parents[cur.v]]$
18:             $key2 \leftarrow cur.keyVert$
19:             if $key1 == key2$ then   ▷ it is just a cycle but not
                    meeting of 2 traversals from diff KeyVertex
20:                 $continue$
21:             $PathMap \leftarrow ComputePathBetween(key1, key2)$
22:             $KeyVertexG[key1].push(key2)$
23:             $KeyVertexG[key2].push(key1)$
24:             $continue$
25:         else $visited[cur.v] == false$
26:             $parents[cur.v] \leftarrow cur.parent$
27:             $keys[cur.v] \leftarrow cur.keyVert$
28:             $visited[cur.v] \leftarrow true$
29:         for all $v \in G[cur.v].adj$ do int v = topology[cur.v][i];
30:             if $dist_s[v] + dist_t[v] > l$ then
31:                 $continue$
32:             $en \leftarrow CompEntropy(cur.keyVert, cur.v) +$
                    $(cur.dist + 1)/l$
33:             $node \ n(v, cur.keyVert, cur.v, cur.dist + 1, en)$
34:             $qu.push(n)$
35:     return $(KeyVertexG, Path)$
```

4.3.2 Candidate Path Search

After executing the stitching algorithm, $KeyVertexG$ and $Path$ are found. The path search algorithm is used to find paths from s to t via key vertices in the key vertex graph - $KeyVertexG$. The actual path are recovered using $path$ after $s - t$ in $KeyVertexG$ are found. Both entropy and distance from s are taken into account in the priority queue. We set priority as $entropy + current \quad distance/l$ if $current \quad distance < l$; otherwise, we set priority as $entropy + current \quad distance$, in order to pennalize path in $KeyVertexG$ that are longer than l.

Algorithm 2 is the pseudo code of the path search algorithm.

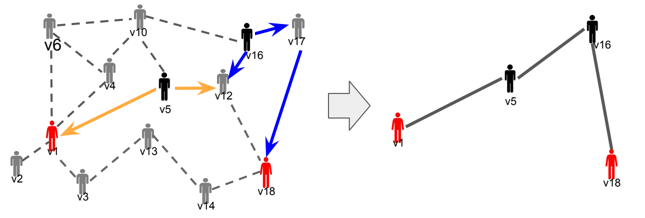


Figure 2: Stitching Algorithm (left) and Candidate Path (right)

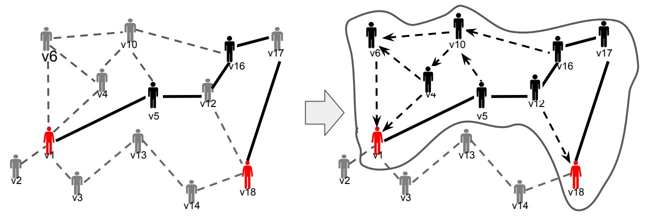


Figure 3: a Candidate Path (left) and a Vertex Group P_i (right)

Algorithm 2 finds shortest path from s to t on $KeyVertexG$ based on entropy value (line 5). After a path p is found, p is removed from $KeyVertexG$ (line 7). The algorithm continues until no more path can be found.

4.3.3 *Conceptual Example*

Figure 2 illustrates the concept of stitching algorithm and candidate path search. Suppose v_5 and v_{15} are key vertices retrieved from the index and v_1 and v_{18} are s and t respectively. v_5 expands to v_1, and edges from v_1 to v_5 and v_5 to v_1 are put into $KeyVertexG$. v_5 also expands to v_{12}. v_{15} expands to v_{17} and v_{12}. When v_{15} expands to v_{12}, v_{12} was occupied by v_5 already. Hence, we can put edges v_{15} to v_5 and v_5 to v_{15} into $KeyVertexG$. After that, v_{15} expands to v_{18} , and edges from v_{15} to v_{18} and v_{18} to v_{15} are put into $KeyVertexG$. Given the $KeyVertexG$, candidate paths from s to t are found based on entropy values, and those candidate paths will be used for path inflation in the next phrase.

Algorithm 2 Path Search Algorithm

```
 1: procedure PATHSEARCH($KeyVertexG, s, t, l, \alpha$)
 2:     $boolean\ \ PathFound \leftarrow true$
 3:     $Array < Path >\ \ CandPath$
 4:     while $PathFound == true$ do
 5:         $p \leftarrow FindShortestPath(KeyVertexG, s, t, l, \alpha)$
 6:         if $p! = \emptyset$ then
 7:             $RemovePath(p, KeyVertexG)$
 8:             $CandPath.push\_back(p)$
 9:         else
10:             $PathFound \leftarrow false$
11:     return $CandPath$
```

4.4 Candidate Path Inflation

After all candidate paths, $CandPath$ are found, the candidate paths are used to form path summary. We developed the path inflation algorithm which greedily includes vertices into path vertex groups.

Algorithm 3 is the pseudo code of the path inflation algorithm. Firstly, all vertices in the $CandPath$ are put into a priority queue which uses entropy as the priority (lines 4-8). Then, the vertex cur in the candidate path with the lowest entropy are popped from the priority queue (line 10). If cur was not visited before, cur is included in the path group $PathSummary[cur.PathID]$ (line 14). After that, all adjacent vertices of cur that satisfy $dist_s[v] + dist_t[v] > l\ or\ (cur.dist + 1) * 3 > dist_s[t]$ (line 19) are pushed into the priority queue with $entropy(cur.P \cup v)$ as cost. $(cur.dist + 1) * 3 > dist_s[t]$ is included so as to prevent vertices that are too far away from vertices in $CandPath$ are included in the path summary. The algorithm terminates when the priority queue becomes empty.

Figure 3 illustrates the concept of candidate path inflation. Given that $v_1 \rightarrow v_5 \rightarrow v_{12} \rightarrow v_{16} \rightarrow v_{17} \rightarrow v_{18}$ is the candidate path. The path inflation algorithm first puts all vertices (i.e. $v_5, v_{12}, V_{16}, v_{17}$) in the candidate path into the priority queue with entropy of the candidate path as priority. Firstly, vertices that are adjacent to $v_5, v_{12}, V_{16}, v_{17}$ are included into the candidate path. Then, other vertices that are adjacent to vertices (e.g. v_4, v_6) in the candidate path are gradually included in the candidate path until the distance constraint. Finally, we will get a subgraph shown in Figure 3 (right).

Algorithm 3 Path Inflation Algorithm

```
 1: procedure PATHINFLATION($G, CandPath, s, t, l$)
 2:     $priority_queue < node > qu$
 3:     $Array < bool > visited$
 4:     for all $p \in CandPath$ do
 5:         for all $v \in p$ do
 6:             $entropy \leftarrow ComputeEntropy(p)$
 7:             $node\ \ n(v, i, v, 0, l, entropy)$
 8:             $qu.push(n)$
 9:     while $!qu.empty()$ do
10:         $cur \leftarrow qu.pop()$
11:         if $visited[cur.v] == true$ then
12:             $continue$
13:         $visited[cur.v] \leftarrow true$
14:         $PathSummary[cur.pathID].push(cur.v)$ ▷ assign
                v into that path group
15:         for all $v \in G[cur.v].adj$ do
16:             if $dist_s[v] + dist_t[v] > l\ or\ (cur.dist + 1) * 3 >$
                    $dist_s[t]$ then
17:                 $continue$
18:             $en = ComputeEntropy(PathSummary[cur.pathID], v)$
19:             $node\ \ n(v, cur.pathID, cur.dist +$
                    $1, cur.l, cur.keyVertex, en)$
20:             $qu.push(n)$
21:     return $PathSummary$                    ▷ return Path Summary
```

5. EVALUATION

All experiments were performed under 64-bit Linux Ubuntu 14.04 on a machine with an Intel 4GHz CPU (4-core), 16 gigabytes of memory, and 1 terabyte solid state drive with 512k block size. All our implementations are in C++ without parallelism.

We first introduce the graph dataset and attributes that we used for the experiments. Then, we present the result of our case studies. Finally, we look at the change of change of path summary quality (i.e. change of entropy) along with the change in the expected number of key vertex l and the number of hints H.

Table 1: Dataset and Parameter

Real Graph	**Num of Vertex**	**Num of Edge**
fb-bfs1 [5]	*1.18m*	*29.78m*
Parameter	**Default**	**Vary**
Exp. Num of Key Vert.	*6*	*3,6,9,12*
Num of Hint	*3*	*1,3,6,12*

5.1 Datasets

We used a real social network dataset fb-$bfs1$ [5], which has 1.63m vertices and 15.14m edges, for our experiments. To control the number of attributes and attribute domain sizes, we generate attributes (Table 2) based on vertex attributes in facebook graph-API [1].

Table 2: Attributes

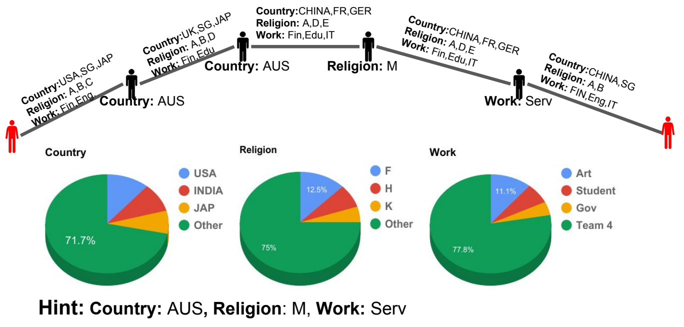
Vertex Attribute	**Domain Size,Distribution** (μ,σ)
AgeGroup	*10, gau(5,2.5)*
Education	*5, gau(3,1.25)*
Gender	*2, uni.*
HomeCountry	*100, gau(50.25)*
Interested in	*3, uni.*
Languages	*50, gau(25,12.5)*
Relationship Status	*2, uni.*
Religion	*20, gau(10,5)*
Work	*50, uni.*
Political	*10, gau(5,2.5)*

5.2 Case Study

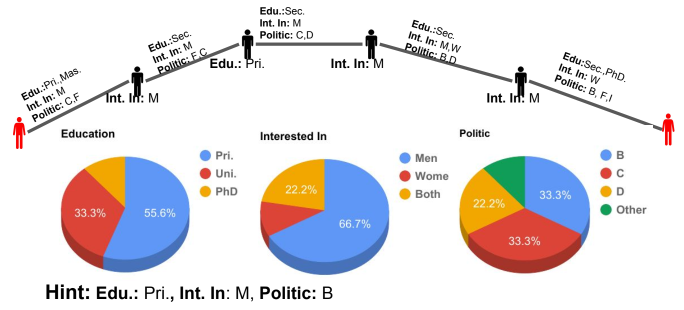
Figure 4 and 5 are four case studies using the $fb-bfs1$ [5] graph. Each of the figures contains a visualization of one of the paths in the path summary. At the top of each figure, we can see the key vertices (man icon) from source to destination. Below each key vertex is the attribute value of the key vertex that matches attribute value in the hint. Attribute value summaries of the path between every two key vertices are shown above the edges between every two key vertices. The pie chats below the path are the summaries of attribute value found by the inflation algorithm. This attribute value summary summarizes the attribute value near to the key vertices.

Case 1: For the first case study in Figure 4(a), we set the expected number of key vertex $l = 6$, the number of hint $H = 3$, and the hint contains attribute $Country = AUS$, $Religion = M$, and $Work = Service$. We can see there are 6 key vertices (including source and destination), which match our expected number of key vertices. Furthermore, the summaries of attribute values on the path between every two key vertices are concise. That gives users a clear idea of attribute values between key vertices. From the pie charts, we can see that $other$ (green) occupies a large portion of the pie. That tells users that attribute values close to the path are inconsistent and probably having large attribute value domain.

Case 2: For the first case study in Figure 4(b), we set the expected number of key vertex $l = 6$, the number of hint $H = 3$, and the hint contains attribute $Education = Primary$, $Interested\ \ In = Men$, and $Politic = B$. We can see there are 6 key vertices (including source and destination), which matches our expected number of key vertices. Furthermore, the summaries of attribute values on the path between every two key vertices are concise. That gives users a clear idea of attribute values between key vertices. From the pie charts, we can see that the top-2 attribute values (blue and red) in each attribute occupies a large portion of the pie. That tells users that attribute values close to the path are consistent and that helps users to efficiently construct their queries.

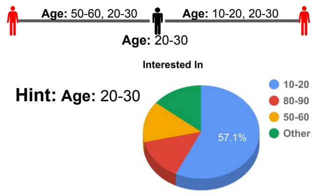


(a) Case 1

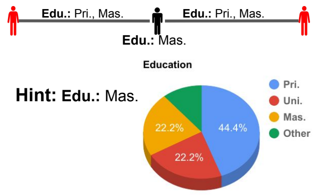


(b) Case 2

Figure 4: Path Summary (Expected Num of Key Vertex=6, Num of Hint=3)



(a) Case 3



(b) Case 4

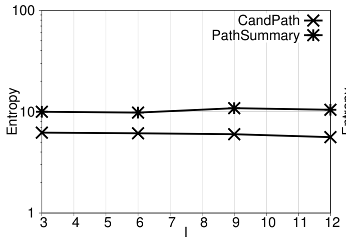
Figure 5: Path Summary (Expected Num of Key Vertex=3, Num of Hint=1)

After we study path summary with 6 key vertices and 3 hints, we try to look at cases with less key vertices and hints.

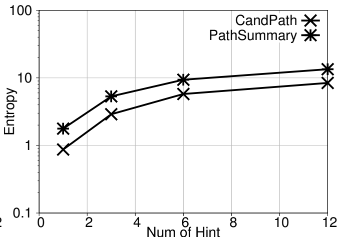
Case 3: For the first case study in Figure 5(a), we set the expected number of key vertex $l = 3$, the number of hint $H = 1$, and the hint contains attribute $Age = 20 - 30$. We can see there are 3 key vertices (including source and destination), which matches our expected number of key vertices. Furthermore, the summaries of attribute values on the path between every two key vertices are also concise. From the pie charts, we can see that the top-1 attribute values (blue) in each attribute occupies a large portion of the pie. On the contrary, the "$other$" attribute value (green) only occupies a small portion. That tells users that attribute values close to the path are very consistent.

Case 4: For the first case study in Figure 5(b), we set the expected number of key vertex $l = 3$, the number of hint $H = 1$, and the hint contains attribute $Education = Master$. We found similar result as in Figure 5(a).

5.3 Query Formulation Using Path Summary



(a)Vary l



(b)Vary Num of Hint

Figure 6: [fb-bfs1] Entropy

In order to connect source and destination via vertices that satisfy attribute hint, we suggest users take into account major attribute values and alternative attribute values when they are formulating queries.

Major Attribute Values: Major attribute values are attribute values that appear on the path between key vertices. For example, in Figure 4(b), "**Edu.**:Sec.,PhD., **Int. In**: W, **Politic**: B,F,I" are major attribute values between destination and the last key vertex. By putting these attribute values into the query, key vertices can be connected. However, based on users preferences, they may not always want to include these major attribute values. Continue with the example in Figure 4(b), users may not want to include "**Edu.:**Sec.,PhD" into the query. If that is the case, users can consider the alternative attribute values.

Alternative Attribute Values: Alternative attribute values are attribute values displayed in the pie charts. They are the distribution of attribute values near to paths between key vertices. Continue with the example in Figure 4(b), if users do not prefer to have "**Edu.**:Sec.,PhD" in the query, they may consider to replace it by "**Edu.**: Uni". Based on the "Education" pie chart, there are 33.3% of vertices has attribute value "**Edu.**: Uni" near to the paths between key vertices. Therefore, conceptually, choosing "**Edu.**: Uni" is similar to rerouting the path between the destination and the last key vertex.

5.4 Change of Entropy

The default expected number of key vertex and number of hint are 6 and 3 respectively. We randomly generate 200 pairs of source and destination and measure the average entropy and execution time.

Figure 6(a) shows the change of entropy along with l. We can see that for both $CandPath$ and $PathSummary$, the entropy does not really increase with l. Although it seems that a longer path would contain more vertices and is more likely to contain different attribute values, this intuition is not supported by Figure 6(a). Since large l offers more opportunity for the algorithm to search for $s - t$ paths with similar attribute values, the increase in path length does not directly imply an increase in entropy.

Figure 6(b) shows the change of entropy along with H. We can see that for both $CandPath$ and $PathSummary$, the entropy increases with H. That is contributed by the fact that more attribute hints mean more attribute are involved, which makes the consistency of attribute values lower.

6. CONCLUSION AND FUTURE WORK

In this paper, we study the problem of computing effective path summary for attributed graphs. We first define a meaningful definition for path summary on attribute graphs that takes into account user's intuition on attribute values as well as path structure properties. Then, we propose an effective 3-phrase algorithm that finds key vertices, stitches key vertices, and searches for path summary. Finally, case studies on the Facebook graph that visualize our path summary results illustrated the effectiveness of our proposed path summary. In the future, we plan to further reduce the number of path in the path summary by proposing the approach that can effectively merge similar paths together so as to further reduce the effort that users need for understanding the path summary.

7. REFERENCES

[1] https://developers.facebook.com/docs/graph-api/reference/user.

[2] S. Cebiric, F. Goasdoué, and I. Manolescu. Query-oriented summarization of RDF graphs. *PVLDB*, 8(12):2012–2015, 2015.

[3] C. Chen, C. X. Lin, M. Fredrikson, M. Christodorescu, X. Yan, and J. Han. Mining graph patterns efficiently via randomized summaries. *PVLDB*, 2(1):742–753, 2009.

[4] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *J. Artif. Intell. Res. (JAIR)*, 1:231–255, 1994.

[5] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proceedings of IEEE INFOCOM '10*, San Diego, CA, March 2010.

[6] I. Herman, G. Melançon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Trans. Vis. Comput. Graph.*, 6(1):24–43, 2000.

[7] K. Khan, W. Nawaz, and Y. Lee. Set-based approximate approach for lossless graph summarization. *Computing*, 97(12):1185–1207, 2015.

[8] Y. Liu, A. Dighe, T. Safavi, and D. Koutra. A graph summarization: A survey. *CoRR*, abs/1612.04883, 2016.

[9] R. Pienta, J. Abello, M. Kahng, and D. H. Chau. Scalable graph exploration and visualization: Sensemaking challenges and opportunities. In *2015 International Conference on Big Data and Smart Computing, BIGCOMP 2015, Jeju, South Korea, February 9-11, 2015*, pages 271–278, 2015.

[10] S. Raghavan and H. Garcia-Molina. Representing web graphs. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 405–416, 2003.

[11] S. Sakr, S. Elnikety, and Y. He. G-SPARQL: a hybrid engine for querying large attributed graphs. In *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 335–344, 2012.

[12] L. Shi, H. Tong, J. Tang, and C. Lin. VEGAS: visual influence graph summarization on citation networks. *IEEE Trans. Knowl. Data Eng.*, 27(12):3417–3431, 2015.

[13] M. Shoaran, A. Thomo, and J. H. Weber-Jahnke. Zero-knowledge private graph summarization. In *Proceedings of the 2013 IEEE International Conference on Big Data, 6-9 October 2013, Santa Clara, CA, USA*, pages 597–605, 2013.

[14] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, pages 567–580, 2008.

[15] Z. Wang, Q. Fan, H. Wang, K. Tan, D. Agrawal, and A. El Abbadi. Pagrol: Parallel graph olap over large-scale attributed graphs. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 496–507, 2014.

[16] Y. Wu, Z. Zhong, W. Xiong, and N. Jing. Graph summarization for attributed graphs. In *2014 International Conference on Information Science, Electronics and Electrical Engineering*, volume 1, pages 503–507, April 2014.

[17] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, pages 880–891, 2010.

[18] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.

[19] Y. Zhou, H. Cheng, and J. X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 689–698, 2010.

Crowd Behaviors Analysis And Abnormal Detection In Structured Scene

Jiaqian Qi, Weibin Liu
Institute of Information Science
Beijing Jiaotong University
Beijing 100044, China
E-mail: wbliu@bjtu.edu.cn

Weiwei Xing
School of Software Engineering
Beijing Jiaotong University
Beijing 100044, China

***Abstract*—Recently, crowd behavior analysis and abnormal trajectory detection have emerged as a significant research field in the visual surveillance. In this paper, we propose a framework for analyzing the crowd behavior and detecting abnormal trajectories in a structured scene. We employ Fast LDA (latent Dirichlet allocation) algorithm to clustering the trajectories. Our framework is based on this method, and it achieves a faster and more accurate clustering result. In order to obtain the crowd motion pattern, we propose two categories of regions of interest, optimal path and critical regions respectively. We use LOF (local outlier factor) algorithm to detect whether the sample points corresponding to the trajectories are abnormal or not, our framework overcomes the problem whose trajectory length is not uniform. Experimental results illustrate that the proposed framework is effective in motion pattern learning and abnormal trajectory detection.**

Keywords-component; Crowd Behavior Analysis; Trajectory clustering; Abnormal Detection; Fast LDA; LOF

I. INTRODUCTION

In recent years crowd behavior analysis and abnormal detection become a challenging topic in the field of crowd management, design of public space, virtual environment, abnormal detection and intelligence environment. In fact, a large number of surveillance devices have been installed in public, such as in markets, subways, gymnasiums, which collect a great deal of pedestrian trajectory data [1], these huge amounts of data is rich in information, which worthy of further study.

Many methods are applied in unstructured scene previously, the approach adopted in those research focuses on the motion patterns learning and abnormal trajectories detection in unstructured situations [2,3,4]. The analysis of crowd behaviors covers different sub problems such as trajectory clustering and trajectory modeling, for which the goal is to automatically learning motion patterns in scene [5]. The task of trajectory clustering is to assign the trajectories with similar measurements to the same cluster. Trajectory modeling is use of parameterized models to represent each trajectory cluster [6]. The task of abnormal detection is to identify those motion behaviors that are significantly different from other moving objects in the same scene, or to distinguish those behaviors with the probability lower than the motion patterns being found before. The task of behavior prediction is to predict the next movement area or semantic behavior patterns of the observed object based on a priori knowledge and the motion patterns of moving objects.

This paper contribute to crowd behavior analysis and abnormal trajectories detection in a structured scene, because of the influence of the road setting and the obstacles in the structured scene, it is difficult to analyze the motion pattern in the structured scene than in the unstructured scene. The crowd behavior analysis and abnormal trajectories detection in the structured scene can be achieved in our framework. First of all, we preprocess the original trajectory dataset, including trajectory segmentation, calculate the motion parameters and encode sub-trajectories respectively. Secondly, we apply Fast LDA (latent Dirichlet allocation) algorithm [7] to cluster the trajectories. In this way, the rough sub-trajectory clusters can be acquired. Thirdly, in order to describe the crowd motion patterns, we propose two categories of regions of interest. Eventually, the LOF (local outlier factor) algorithm is adopt to judge whether the sample points of the trajectories are abnormal, which overcome the problem whose trajectory length is not uniform. The framework is minutely described in Fig. 1.

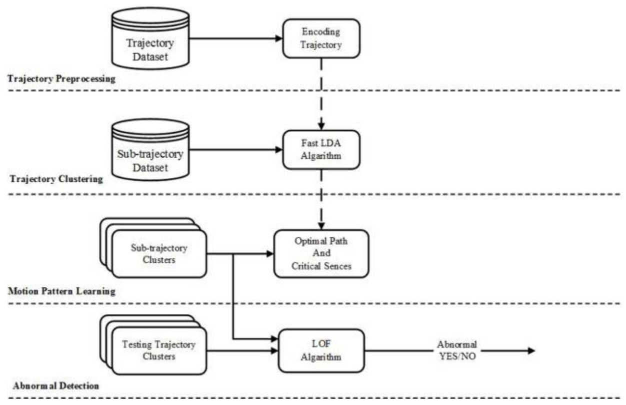


Figure 1. The framework of crowd analysis and abnormal detection.

DOI number: 10.18293/DMSVLSS2017-003
This research is partially supported by National Natural Science Foundation of China (No. 61370127, No.61473031, No.61472030), Program for New Century Excellent Talents in University (NCET-13-0659), Fundamental Research Funds for the Central Universities (2014JBZ004), Beijing Higher Education Young Elite Teacher Project (YETP0583). The opinions expressed are solely those of the authors and not the sponsors.
{Corresponding author: Weibin Liu, wbliu@bjtu.edu.cn }

The remainder of the paper is organized as follows: Section 2 presents the relate work. Section 3 reviews the detail of crowd behavior analysis, which contain trajectory preprocessing, sub-trajectory clustering and motion pattern learning respectively. In section 4, we presents the detail of abnormal trajectories detection. Experiments are presented in section 5.

II. Relate Work

Motion patterns learning devoted to build the regular to expression the motion trajectories by the observed data. In [8], Zou et al propose a new approach which constructed c Latent Dirichlet Allocation (CLDA) model to do trajectory clustering. In [9], Zou et al proposed a belief based correlated topic model (BCTM) to do trajectory clustering. Hu et al. [6] clustered the trajectories by using of Dirichlet process mixture model (DPMM) clustering algorithm. Modeled the trajectories by use of a time-sensitive Dirichlet process mixture model (tDPMM).

Abnormal trajectory detection is to identify those motion behaviors that are significantly different from other moving objects in the same scene, or to distinguish those behaviors with the probability lower than the motion patterns being found before. In [10], Claudio Rosito Jung et al proposed to 4-D histogram to detect the abnormal trajectory. In [11], Serhan Cosar proposed an integrated method that incorporates the trajectory-based analysis and pixel-based analysis for abnormal behavior inference.

Behavior prediction is to predict the next movement area or semantic behavior patterns of the observed object based on a priori knowledge and the motion patterns of moving objects. In [12], Josh Jia-Ching Ying et al combined the geographic features and the semantic features to detect the abnormal trajectory.

III. Crowed Behavior Analysis

In the following, we first preprocess the trajectory dataset, then cluster the sub-trajectory, finally we learn the crowd motion patterns.

A. Preprocessing trajectory dataset

1) trajectory segmentation:

The preprocessing of trajectories mainly contains three steps in our framework: trajectory segmentation, calculate the motion parameters and encode sub-trajectory.

For charactering spatiotemporal information about the trajectory, we use the sequence of flow vectors to represent the trajectory, In order to stabilize the movement trend of trajectory, a trajectory is segment into a series of sub-trajectory at the turning point. In our case, motion direction can be divided into eight directions: east (0), northeast (1/4π), north (1/2π), northwest (3/4π), west (π), southwest (5/4π), south (3/2π), southeast (7/4π). Then the motion direction of each sample point are discretized to the approximate standard direction. If the discretized motion direction of a sample point is different from the previous, these points are regarded as a turning point of the trajectory. Then the trajectories of moving objects are segmented at the turning points.

$$F_n = \{F_1', F_2', \dots, F_m'\}$$
$$F_i' = \{f_1, f_2, \dots, f_j\} \qquad (1)$$

The mathematical expression of the sub-trajectory shown as formula (1), where F_n is nth trajectory in the trajectory database, m is the number of sub-trajectories of trajectoryF_n, j is the number of sample points in the ith sub-trajectory, namely the length of the sub-trajectory. The moving direction and motion tendency of each sub-trajectory is relatively stable. The sub-trajectories are easier to researched and analyzed than the whole trajectories.

2) Obtain the motion parameters:

From the above discussion a sub-trajectory in dataset are defined as a sequence of flow vector $F_i' = \{f_1, f_2, \dots, f_j\}$, where, $f_j = \{x^t, y^t, v^t, \theta^t\}$, original dataset contains only location information x^t, y^t, so the value of velocity and direction of f_j at time t should be calculated. So the unknown and useful motion parameters from the original dataset should be extracted for further trajectory processing. The value of velocity v^t and value of direction θ^t of f_j can be calculated by use of formula (2) and (3).

$$v^t = \sqrt{(\Delta x)^2 + (\Delta y)^2} \qquad (2)$$

Where

$$\Delta x = \begin{cases} 0 & t = 1 \\ x^t - x^{t-1} & t \le j \end{cases}$$

$$\Delta y = \begin{cases} 0 & t = 1 \\ y^t - y^{t-1} & t \le j \end{cases}$$

$$\theta^t = \begin{cases} \frac{tan^{-1}(\Delta y)}{\Delta x}, & if\ (\Delta x) > 0 \\ \frac{tan^{-1}(\Delta y)}{\Delta x}, & if\ (\Delta x) < 0\ and\ (\Delta y) \ge 0 \\ \frac{tan^{-1}(\Delta y)}{\Delta x}, & if\ (\Delta x) < 0\ and\ (\Delta y) < 0 \\ \frac{\pi}{2}, & if\ (\Delta x) = 0\ and\ (\Delta y) > 0 \\ -\frac{\pi}{2}, & if\ (\Delta x) = 0\ and\ (\Delta y) < 0 \\ 0, & if\ (\Delta x) = 0\ and\ (\Delta y) = 0 \end{cases} \qquad (3)$$

3) Encode sub-trajectories:

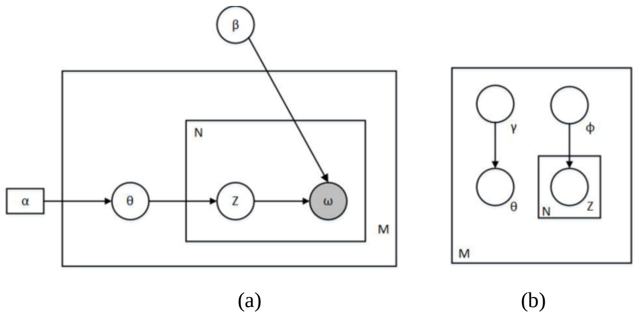
LDA [13] usually used to cluster co-occuring words into one topic, it can't cluster trajectories directly, for further analysis by Fast LDA, it necessary to encode the trajectories and map the sequence of flow vector of trajectories into words in a codebook [3]. We quantized the trajectories according to a codebook. First, we define a codebook to encode the sub-trajectories, we divide the scene image into cells of 10×10 pixels, so the space of the scene uniformly quantized into small cells, and quantize the moving direction of each trajectory point into 5 bins, as$\theta \in \{0,1,2,3,4\}$. The resolution of the space of the scene is supposed to be$M(pixel) \times N(pixel)$, for any sequence of trajectory $x^t \in \{1,2,\dots,M\}$ $y^t \in \{1,2,\dots,N\}$. According to the codebook, the scene are segmented into $(M/(\frac{M}{10})) \times (N/(\frac{N}{10}))$ rectangular areas, the size of each rectangular is 10×10 pixels. Each rectangle area is represented by the direction of trajectory, 0 indicates without trajectory. Then the directions can be discretized into 5 bins by formula (4)

$$\theta_d^t = \begin{cases} 1, & \frac{\pi}{4} < \theta^t \le \frac{3\pi}{4} \\ 2, & \frac{-\pi}{4} < \theta^t \le \frac{\pi}{4} \\ 3, & \frac{-3\pi}{4} < \theta^t \le \frac{-\pi}{4} \\ 4, & \frac{3\pi}{4} < \theta^t \le \pi \text{ or } -\pi < \theta^t \le \frac{-3\pi}{4} \end{cases} \quad (4)$$

Then the codebook can be defined as a set of code values:

$$\text{codebook} = \{0,1,2,3,4\} \quad (5)$$

B. Sub-trajectory Clustering



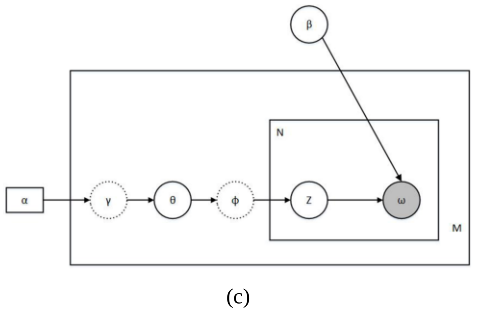


Figure 2. Fast LDA . (a) Graphcial representation of LDA [13]; (b) Varitional distributions for fast model[7]; (c) Fast varitional inference of LDA.

To make the paper independent, we first review the LDA algorithm. In Fig. 2, (a) shows the graphical representation of LDA [13]. In this basic model, shaded circles represent the observed variables, unshaded circles represent latent variables. The arrows indicate dependencies between two variables. Rectangle represents repeated sampling, the number of repetitions in the lower corner. θ is a vector of topic, which is follow a Dirichlet distribution which parameter is α. Z represented a topic, which is a latent variable. ω represents words. In our case, M is number of clusters. N is the number of trajectory. In Fig.2, (b) shows the varitional distributions for fast model [7], (c) illustrate fast varitional inference of LDA. γ and ϕ are the intermediate parameter. LDA has two model parameters α and β: α is the parameter of Dirichlet distribution, and β is the set of the discrete distribution parameters for each of k components over V words, where k is the dimensionality of the Dirichlet distribution is assumed known and fixed [13], where V is the size of the dictionary. Fast LDA algorithm use fast variational inference to estimate the parameters α and β, in our experimental, we randomly initialized α and β.

After encoding, a sub-trajectory is represented by a sequence of words in codebook, so all of sub-trajectory treated as a document, each of the sub-trajectory treated as a word, we use Fast LDA to find out the latent topics, assign the sub-trajectories with similar measurements to the same topic.

C. Crowd pattern learning

In this paper, we propose two kinds of ROI (regions of interest) as crowd motion pattern. The description is as follows:

1) Optimal path:

The optimal path of a trajectory clusters are these scene areas, which contain many sample points of the clusters, corresponding to the region where moving object through with high probability. In our paper, we propose that the optimal path contributes to the anomaly trajectory detection and propose some suggestions for design of public space.

Assuming that the pixel of the scene picture is $\mathrm{m} \times \mathrm{n}$, then the scene can be divided into $(\mathrm{m}/\mathrm{m}/10) \times (\mathrm{n}/\mathrm{n}/10)$ small regions, such as in Fig. 3 (b). If a region contains more than a certain number of sample points it can be considered as optimal path such as Fig. 3 (c). We set different thresholds to divide the region of interest into several grades with different shades of color by formula (6), the more sampling points in this area, the more important the area is.

$$\text{each block} = \begin{cases} l & \text{sample points} < \mathrm{a} \\ m & a \le sample\ points \le b \\ h & sample\ points > b \end{cases} \quad (6)$$

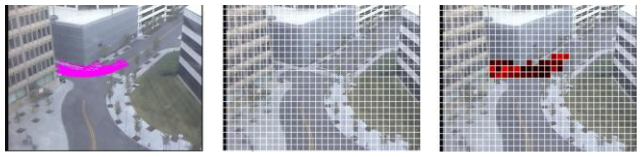


Figure 3. The framework of crowd analysis and abnormal detection. (a) A sub-trajectory cluster; (b) Scene block; (c)Optimal path.

2) Critical regions of scene:

There has many obstacles and facility in the structured scene, so the trajectories of moving objects will more easily changed than unstructured scene. These regions are significant meaningful to design of public space, public management and public security, and can be called the critical regions of scene. There are two types of critical regions in structured scene:

a) Motion trend change region:

The regions where moving objects change their motion tendency in whole structured scene is named the motion trend change region. Assume that F_n is nth trajectory in the trajectory database, then F_n can be represents with a sequence $F_n = \{f_1, f_2, \dots, f_m\}$, where m is the number of sample points in the nth trajectory, $f_m = \{x^t, y^t\}$. The turning point can be calculated by formula (3) and (4). Then formula (6) are used to count the number of sample points in each regions.

b) Densely region:

The regions where moving objects are often gathered are called densely region. The scene is divided into 10×10 pixels rectangle firstly, then number of sample points in each rectangle and in whole scene can be calculated, the formula (7) are used to judge whether the region is densely region. The formula is as follows, where, $N_{rectangle}$ represent the number of sample points in each rectangle, $N_{whole\ scene}$ represent the number of sample points in whole scene, the D represent the ratio of the two:

$$\mathrm{D} = \frac{N_{rectangle}}{N_{whole\ scene}} \tag{7}$$

IV. Abnormal trajectory detection

A. Testing Trajectories Preprocessing

The testing trajectory dataset needs to be preprocessed before the abnormal trajectory detection for reducing computation. We segmented the original trajectories, calculated the parameters of sub-trajectories and encoded sub-trajectories. Only preprocessing is not enough, each testing sub-trajectory should be assigned to the different cluster according to training clusters. When detecting abnormal trajectories, each testing sub-trajectory just need to compare with the most similar training sub-trajectory cluster rather than the whole, experiments show that it is rapid and effect.

B. Abnormal Trajectory Detecting

The outliers in LOF algorithm [14] are those objects that abnormal in the local scope rather than in global perspective. In traditional methods, objects are only two states: normal and abnormal, so the traditional concept of outliers is simple rigid property. However, LOF algorithm use local outlier factor to describe the degree of the outlier of objects, so it is more applicable to the related applications in real life. LOF algorithm is to identify the abnormal objects by calculating the degree of outlier of each point. The steps of LOF algorithm for abnormal detection is as follows:

1) $k - distance$ of an object p:

Assuming that k is a positive integer, p and o are sample point on testing trajectory, the distance between p and object o is defined as follows, it satisfies two requirements:

(I) at least k objects $o' \in D\{p\}$, it conform that $\mathrm{d}(\mathrm{p}, o') \leq \mathrm{d}(\mathrm{p}, \mathrm{o})$

(II) at most k-1 objects $o' \in D\{p\}$, it conform that $\mathrm{d}(\mathrm{p}, o') < \mathrm{d}(\mathrm{p}, \mathrm{o})$;

2) $N_{k-distance}$ of object p :

The $\mathrm{k-distance}$ of p has known, then calculate $N_{k-distance}$ of object p on the testing trajectory, the definition of $N_{k-distance}$ is as follows:

$$N_{k-distance(p)} = \{q | d(p,q) \leq k - distance(p)\} \tag{8}$$

3) Reachability distance of object p and o:

Then we calculate the reachability distance of object p and o on the testing trajectory. For any natural number k. The reachability distance of object p and object o can be defined by the following formula:

$$\mathrm{reach} - dist_k(p, o) = \max\{distance(o), d(p, o)\} \tag{9}$$

4) Local reachability density of an object p:

The local reachability density of an object p on the testing trajectory should be calculated after obtained the value of reachability distance. The definition of local reachability density of p is as follows:

$$lrd_{MinPts}(p) = \frac{1}{\left[\frac{\sum_{o \in N_{MinPts}} reach-dist_{MinPts}(p,o)}{|N_{MinPts(p)}|}\right]} \tag{10}$$

5) Local outlier factor of an object p:

Finally, we calculate the local outlier factor of an object p on the testing trajectory. The local outlier factor of the sample object p can represent the local anomaly possibility of the sample points, it described as follows:

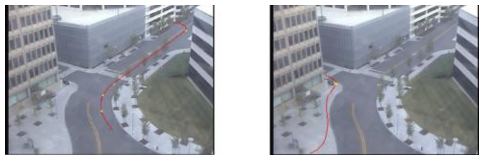
$$LOF_{MinPts(p)} = \frac{\sum_{o \in N_{MinPts(p)}} \frac{lrd_{MinPts}(o)}{lrd_{MinPts}(p)}}{|N_{MinPts(p)}|} \tag{11}$$

If the local outlier factor of an object p on the testing trajectory is large than certain value, point P can be marked as an abnormal sample point. And if most of the sample points of a sub-trajectory are abnormal, this sub-trajectory can be regarded as an abnormal sub-trajectory.

V. Experiment Results

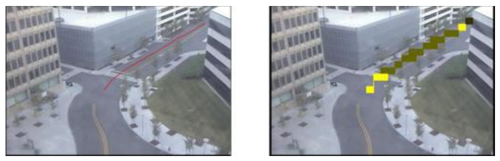
Experiments are conducted on the pedestrian trajectory database collected from the Edinburgh University [15,16]. This section will demonstrate the results of experiment based on the previous work. The resolution of the space of the scene is 640×480, according to the encoding sub-trajectory in section 3, the codebook size is 32×24, so the feature dimensionality of sub-trajectory is 768.

A. Trajectory processing



(a) (b)

Figure 4. Trajectory segmentation. (a) and (b) are two original trajectory respectively.



(a) (b)

Figure 5. (a) A sub-trajectory before encode; (b) encode result.

The trajectories are segmented into several sub-trajectories at their critical points, which show as Fig. 4, we selected two trajectories randomly, where the yellow point represents the critical point of the trajectory, which is the turning point of the trajectory. Each sub-trajectory F_i' can be encoded just as Fig. 5, the left figure show a sub-trajectory before encoding, the right figure is corresponding coding results, and the different colors of rectangular represent different directions of trajectory.

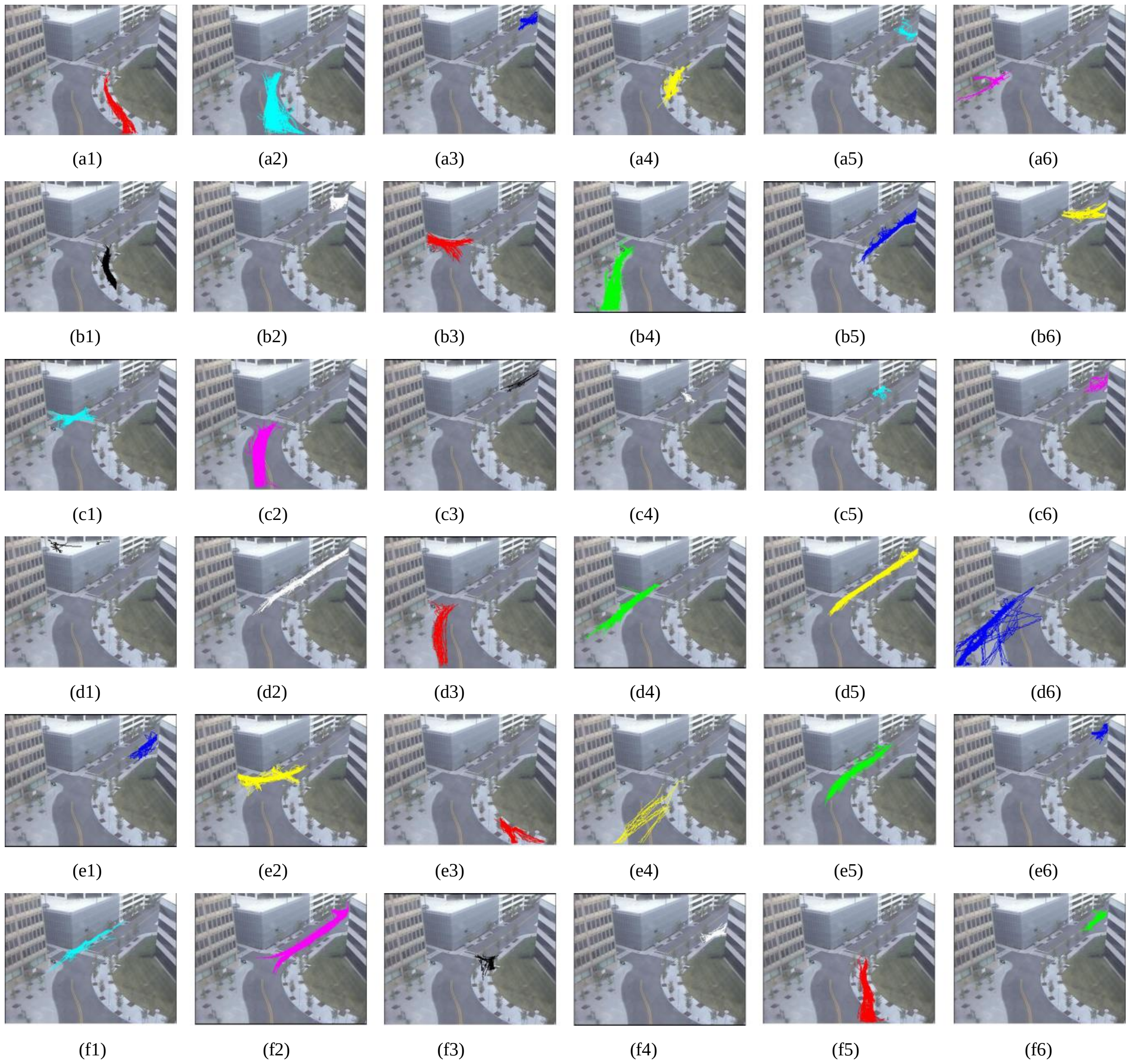

(a1) (a2) (a3) (a4) (a5) (a6)
(b1) (b2) (b3) (b4) (b5) (b6)
(c1) (c2) (c3) (c4) (c5) (c6)
(d1) (d2) (d3) (d4) (d5) (d6)
(e1) (e2) (e3) (e4) (e5) (e6)
(f1) (f2) (f3) (f4) (f5) (f6)

Figure 6. Sub-trajectory clusters. (a1)-(a6), (b1)-(b6), (c1)-(c6), (d1)-(d6), (e1)-(e6), (f1)-(f6) are 36 sub-trajectory clusters.

B. *Sub-trajectory clustering*

After trajectory preprocessing, a sub-trajectory training dataset has been obtained. All the sub-trajectories are clustered into several clusters by the Fast LDA clustering algorithm. Fig.6 illustrates the result of clustering by Fast LAD algorithm, then 36 sub-trajectory clusters in the structured scene can be obtained, each of them represents a kind of crowd motion pattern in a structured scene. Most of the sub-trajectory clusters go along the prescribed path such as both sides of the road or zebra crossing, such as a(1), a(2), a(4),b(1), c(2) and so on, the other sub-trajectory clusters not belong to the prescribed path needs to be study in intensive, such as a(5), b(6), e(4).

Clustering validation has been considered an important indicator for the success of clustering applications [17]. In general, clustering validation consists two classes, external and internal clustering validation [17]. In this paper, we use internal clustering validation to judge the success of the clustering. SSE (sum of the squared errors) and S_Dbw (summation density between and within) [17] are internal clustering validity index respectively, the minimum value of those two index indicates the optimal cluster result. We compared the clustering trajectory validation of our algorithm with those of two clustering algorithms: K-means clustering [18], clustering algorithm based on max-min distance [19]. In experiment, we selected the appropriate parameters of the two comparison algorithms, in order to make the clustering results as precisely as possible.

Table I illustrates comparison of SSE from three approaches, it can be seen that our approach can obtain the minimum value of SSE. Table II shows that comparison of S_Dbw from three approaches, compared with the other two methods, our method can obtain the minimum value of S_Dbw. Therefore our method has better clustering performance than the other two methods.

TABLE I. SSE (SUM OF THE SQUARED ERRORS)

	Number of cluster:					
	26	***37***	***42***	***46***	***54***	***57***
Fast LDA clustering	7.58	7.09	6.04	6.80	6.91	6.96
K-means clustering	17.14	12.29	18.60	19.97	20.96	24.53
max-min distance	18.14	14.29	18.60	19.97	21.96	23.53

TABLE II. S_DBW (SUMMATION DENSITY BETWEEN AND WITHIN)

	Number of cluster:					
	26	***37***	***42***	***46***	***54***	***57***
Fast LDA clustering	1.36	1.50	1.64	2.17	2.76	2.96
K-means clustering	4.60	4.94	5.66	5.77	6.51	6.74
max-min distance	5.60	5.94	6.66	6.77	6.91	7.34

C. *Crowd motion behaviors analysis*

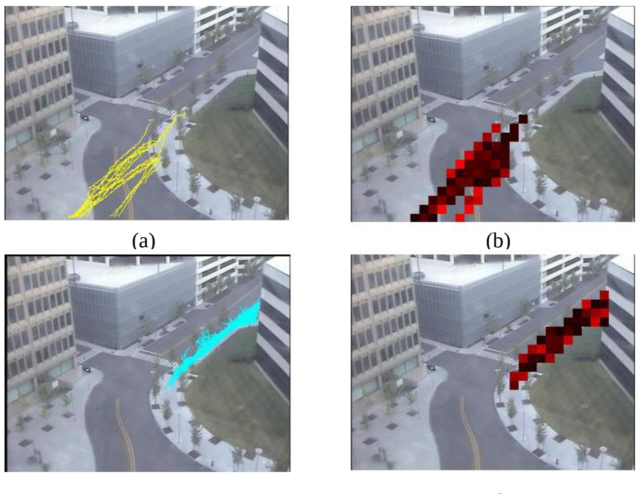

(a) (b) (c) (d)

Figure 7. Optimal path of cluster. (a) Cluster34; (b) Optimal path of cluster 34; (c) Cluster11; (d) Optimal path of cluster 11.

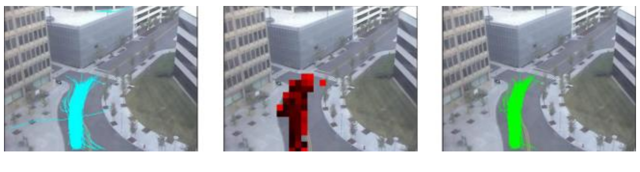

(a) (b) (c)

Figure 8. Refining sub-trajectory cluster by optimal path. (a) Cluster14; (b) Optimal path of cluster14; (c) Result of refined.

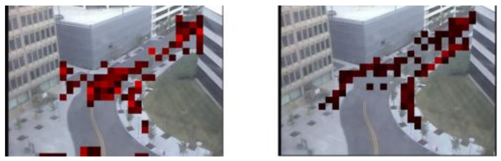

(a) (b)

Figure 9. The critical regions of sence. (a) The motion trend change regions; (b) The densely regions.

The optimal path of each sub-trajectory cluster can describe the areas in the structured scene where attractive most pedestrian in this cluster, the optimal path can provide some suggestions for the redesign of the scene. Just as show in Fig.7, many pedestrians in (a) pass through the road not along original design in scene, so the regions of interest will be useful when the scene need to redesign, the designer can set pedestrian crosswalk at those regions in (b). Even though the pedestrians in (c) move along the original design, there are also some areas where trajectories too intensive, more cameras should be set at the regions in (d).

In Fig.8, (a) represents a sub-trajectory cluster, and (b) is the corresponding optimal path. The optimal path of each sub-trajectory cluster is not only helpful to obtain the information of the crowd motion behaviors and structure scene, but also can make a refining to each sub-trajectory cluster. If a sub-trajectory in a sub-trajectory cluster does not pass through the optimal path, it means that the space feature distribution of this sub-trajectory is low probability in its cluster and it can be removed.

Fig. 9 (a) shows the motion trend change regions. They often locate at the road turning. Motion trend change regions illustrate the turning point of trajectory, which can used in the design of public space. Densely regions illustrate the regions whose density is higher than the others. Just as in figure (b). The depths of the color also show the degrees of the attention to the regions. They often located at the building entrance and exit, more cameras should be installed in this place.

D. *Abnormal detection*

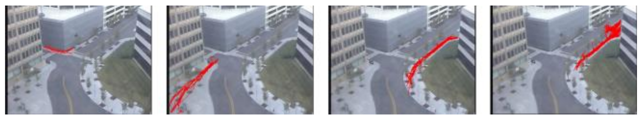

(a) (b) (c) (d)

Figure 10. Classifying the testing sub-trajectories. (a)-(d) are four testing sub-trajectory.

According to the motion behaviors information of the crowd in a structured scene achieved before, the LOF algorithm is apply to detect the abnormal sub-trajectories. We segmented the testing trajectory dataset into sub-trajectory dataset and then grouped them into several testing sub-trajectory clusters, which can reduce calculation complexity. In Fig.10, we demonstrate the part of the testing trajectory clusters.

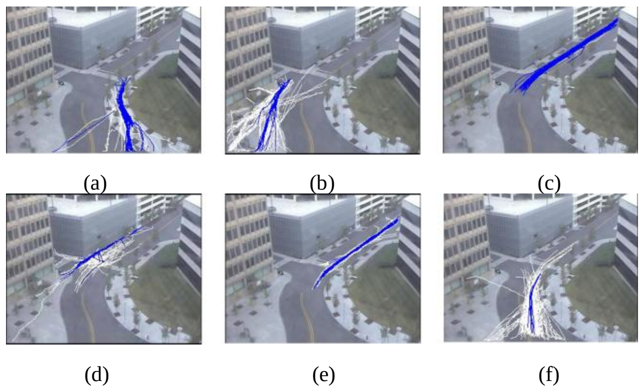


Figure 11. Abnormal detection. (a)-(f) are 6 results of abnormal detection

In Fig. 11, we demonstrate the results of abnormal detection of 6 clusters. The blue sub-trajectories are normal trajectory and the white sub-trajectories are abnormal trajectory. Experimental results show that LOF algorithm could detect abnormal trajectory in testing trajectory dataset. It could find the objects which away from most other moving objects in a certain period of time. The white trajectory significantly deviate from the blue trajectory. The trajectory of moving object like the white trajectory need further observation and analyze, to ensure the safe and effective of observation scene.

VI. Conclusion

In this paper, we proposed a framework to analyze the crowd behaviors and detect abnormal trajectories in a structured scene. Fast LDA algorithm is adopt to clustering sub-trajectory. All of sub-trajectory treated as a document, each of the sub-trajectory treated as a word, and the sub-trajectory clusters can be treated as topics. The clustering result compared with two other clustering algorithms, which proved that our method has good performance. In order to describe the crowd motion patterns of each trajectory clusters, we propose two categories of regions of interest, optimal path and critical regions respectively, which can propose some suggestions for design of public space. In the last part, we use LOF algorithm to detect abnormal trajectory, which overcome the problem that the trajectory length is not uniform. Experimental results demonstrate the good performance of our proposed framework. There are still some improvements in our research, for example, the trajectory prediction, which is an important research direction. In the future research, we will focus on the trajectory prediction.

References

[1] Peter Widhalm, Norbert, “Learning Major Pedestrian Flows in Crowed Scenes,” Pattern Recognition, International Conference on, vol. 00, no. , pp. 4064-4067, 2010.

[2] Jing Cui, Weibin Liu, “Crowd behaviors analysis and abnormal detection based on surveillance data,” Journal of Visual Languages and Computing, v 25, n 6, 628-36, Dec. 2014.

[3] Xinyi Chong, Weibin Liu, “Hierarchical crowd analysis and anomaly detection,” Journal of Visual Languages and Computing, v 25, n 4, 376-93, Aug. 2014.

[4] Weibin Liu, Xinyi Chong, “Learning motion patterns in unstructured scene based on latent structural information,” Journal of Visual Languages and Computing, v 25, n 1, 43-53, Feb. 2014.

[5] Hajer FRADI, Bertrand LUVISON and Quoc Cuong PHAM, “Crowd Behavior Analysis Using Local Mid-Level Visual Descriptors,” IEEE Transactions on Circuits and Systems for Video Technology, PP(99):1-1 • October 2016.

[6] Weiming Hu, Xi Li and Guodong Tian, “An Incremental DPMM-Based Method for Trajectory Clustering, Modeling, and Retrieval,” IEEE transactions on pattern analysis and machine intelligence, VOL. 35, NO. 5, MAY 2013.

[7] Hanhuai Shan, Arindam Banerjee, “Mixed-membership naive Bayes models” in Data Mining and Knowledge Discovery, Vol 23, 1-62, (2011).

[8] Jialing Zhou, Yanting Cui and Fang Wan, “A cluster specific latent Dirichlet allocation model for trajectory clustering in crowded videos,” IEEE International Conference on Image Processing (ICIP), 2348-52, 2014.

[9] Jialing Zou, Qixiang Ye, Yanting Cui, “A Belief based Correlated Topic Model for Trajectory Clustering in Crowded Video Scenes, ” IEEE International Conference on Pattern Recognition (ICPR), 2543-8, Aug. 2014.

[10] C.R. Jung, L. Hennemann and S.R. Musse, “Event detection using trajectory clustering and 4-D histograms,” IEEE Trans. Circuits Syst. Video Technol. 18 (November) (2008) 1565–1575.

[11] Serhan Cosar, Giuseppe Donatiello and Vania Bogorny, “Toward Abnormal Trajectory and Event Detection in Video Surveillance,” IEEE transactions on circuits and syster for video technology, Vol. 27, NO. 3, MAR 2017.

[12] J.J. Ying, W.C. Lee, T.C. Weng, V.S. Tseng, “Semantic trajectory mining for location prediction,” ACM SIGSPATIAL GIS'11, Chicago, November 2011, pp. 34–43.

[13] David M. Blei, Andrew Y. Ng, “Latent Dirichlet Allocation,” Journal of Machine Learning Research, Vol. 3, 993-1022 (2003).

[14] Markus M.Breuning, Hans-Peter Kriegel and Raymond T.Ng, “LOF:Identifying density-based local outliers,” Proceedings of the ACM SIGMOD International Conference on Management of Data, 93-104 (2000).

[15] B. Majecka, Statistical models of pedestrian behaviour in the forum,School of Informatics, University of Edinburgh, 2009 (Master0sthesis).

[16] R. Fisher, Edinburgh Informatics Forum Pedestrian Database, 〈 http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/ 〉, 2010.

[17] Yanchi Liu, Zhongmou Li and Hui Xiong, “Understanding of Internal Clustering Validation Measures,” IEEE International Conference on Data Mining, 911-916 (2010).

[18] Xinmin Tang, Junwei Gu, Zhiyuan Shen, “ A Flight Profile Clustering Method Combining Twed With K-Means Algorithm for 4D Trajectory Prediction,” Integrated Communication, Navigation and Surveillance Conference (ICNS), S3:1-9, 2015.

[19] Visalakshi, N. Karthikeyani and J. Suguna, “K-Means Clustering using Max-min Distance Measure ,” Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS), June 14, 2009 - June 17, 2009.

Assessing RDF Graph Databases for Smart City Services

Pierfrancesco Bellini, Paolo Nesi
Distributed Systems and Internet Technology Lab, DISIT, http://www.disit.org
Department of Information Engineering, DINFO, University of Florence, Florence, Italy
pierfrancesco.bellini@unifi.it, paolo.nesi@unifi.it

***Abstract* — RDF stores may be used to set up knowledge bases integrating heterogeneous information for web and mobile applications to use the data for new advanced services to citizens and city administrators, thus exploiting inferential capabilities, temporal and spatial reasoning, and text indexing. In this paper, the needs and constraints for RDF stores to be used for smart cities services, together with the currently available RDF stores are evaluated. The assessment model allows a full understanding of whether they are suitable as a basis for Smart City modeling and application. The comparison of the RDF stores addressed a number of well-known RDF stores. The paper also reports the adoption of the proposed *Smart City RDF Benchmark* on the basis of Florence Smart City model, data sets and tools accessible as Km4City http://www.km4city.org, and adopted in the European Commission international smart city projects named RESOLUTE H2020, REPLICATE H2020, and in Sii-Mobility National Smart City project in Italy.**

Keywords— smart city; RDF stores; graph databases; RDF benchmark; linked data benchmark.

I. Introduction

Smart cities produce large amount of data having a large variability, variety, velocity, and size; and thus complexity. The variety and variability of data can be due to the presence of several different formats, [1], [2] and to the interoperability among semantics of the individual fields and of the several data sets [3]. Static data are rarely updated, for instance once per month/year, which is quite the opposite with dynamic data: they can be updated from once a day up to every minute so as to get real time data. The data velocity is related to the frequency of data update for dynamic data such as position of buses, people flow status, position of waste collectors, etc. The size grows over time accumulating new data every day and week. At architectural level, smart city solutions typically adopt n-tier architectures [4].

The usage of RDF stores in the application domain of Smart City is quite recent, since in most cases services are vertically provided. For example the Intelligent Transport System, ITS, in the city provides information regarding the location of buses and their delay, without addressing the location of city services, flow of people, real time events in the city. Some city data integrators are well-known services such as bike and car sharing, navigator system, tourism information, hotel booking, etc. All these solutions have the need to integrate geo-located information with real time data and events continuously arriving from updated information such as: events, votes, traffic flows, comments, etc. [2], [5]. As to these applications, RDF stores may be a solution to allow addressing the variability of data, so as to make reasoning on space, time, and concepts [6]. The Resource Description Framework specified by W3C allows the representation of facts using "triples" of the form (*subject*, *predicate*, *object*) where URIs are used to identify the entities and the predicates connecting them. Thus a triple represents the arc of the graph connecting two entities and the predicate describes the kind of relation between the two entities. Moreover the object part of the triple can also be a low level data type as string, dates, integers etc. to describe not only the relations among entities but also specific information about them (e.g. name, email, birth date). RDF stores allow storing these triples and the SPARQL query language allows querying them. Some RDF stores can also manage set of triples as a single graph identified by an URI, in this way information about this graph can be provided using other triples (where the subject is the graph itself).

For the evaluation of RDF stores specific assessment models and benchmarks have to be adopted. For example, the LUBM benchmark [7] uses a synthetic dataset in the university domain and covers only the SPARQL 1.0 specification. On the contrary, the BSBM benchmark [8] generates a synthetic dataset in the e-commerce domain and covers the SPARQL 1.1 business analytics queries. More recently, in the Linked Data Benchmarks Council project[1] two benchmarks were proposed both generating a synthetic dataset, one from the semantic publishing domain (LDBC-SP) and the other from the social networks domain (LDBC-SN). The GeoSPARQL standard has been developed by the Open Geospatial Consortium to cover spatial searches, while not many solutions currently support this specification. Regarding the benchmark of geo and spatial RDF stores the Geographica benchmark [9] was proposed by using both a synthetic generated dataset and a real dataset. It analyses the support and performance for advanced spatial relationships among complex spatial entities (e.g., polygons). In [10], the real and synthetic benchmark datasets have been compared showing that synthetic generated datasets are similar to datasets generated for relational database benchmarks (TPC-H) and strongly different from real-world datasets (e.g., dbPedia) being much less structured. In [11], with the SPARQL Performance Benchmark (SP2Bench) a language-specific benchmark framework designed for the most common SPARQL constructs has been proposed.

DOI reference number : 10.18293/DMSVLSS2017-008

[1] http://ldbcouncil.org

Recently SPARQL has been extended to query real-time data coming from RDF data streams. There are some implementations as C-SPARQL [12], SparqlStream [13], CQELS and also specific benchmarks were defined as SRBench [14] using data from weather sensors, LSBench [15] using data from social networks and CityBench [16] using data from smart city sensors. Those kinds of specific benchmark are suitable for streaming data, with queries performing specific requests with limited number of results. W3C also reviewed RDF store benchmarks 2 highlighting their applicability in assessing different aspects of the RDF stores, and their application on different stores.

Despite this wide state of the art on RDF stores benchmarks, none of the mentioned approaches is specifically suitable for assessing the RDF stores against Smart City. Smart City presents extremely particular and specific conditions exploiting the latest and most challenging constructs of the RDF stores as geo-spatial queries, text queries, time queries and combinations of them. On this regard, in this paper, a Smart City RDF Benchmark, SCIRB, has been.

This paper reports the formalization of the proposed *Smart City RDF Assessment Model and Benchmark* and its adoption in comparative assessment of a number of RDF stores. The data and queries adopted for replicating the mentioned assessment have been published on the following web page http://www.disit.org/smartcityrdfbenchmark. The dataset is real and is based on Florence Smart City which in turn is grounded on Km4City ontology and model [3].

The paper is structured as follows. In **Section II**, the major smart city requirements/demands in modeling and accessing semantic knowledge are reported. The requirements can be used as drivers for features based selection of RDF stores. **Section III** presents the general evaluation methodology for assessing and selecting the RDF stores for smart city applications. In **Section IV,** the comparison of most relevant state of the art RDF stores is reported on the basis of the model identified in Section III.A. **Section V** reports the application of the proposed smart city benchmark in assessing the most featured RDF stores (i.e., Virtuoso, GraphDB, Oracle and StarDog). The analysis has highlighted several interesting aspects connected to the performance of RDF stores in: loading and indexing triples, and in performing geographical and textual queries, also during store updates. Conclusions are drawn in **Section VI.**

II. Smart City Requirements for RDF Stores

When providing services to citizens of a smart city, an RDF/graph store should provide some features that allow the support of specific functionalities. In particular, the following features are reported according to their relevance and classifying them. Therefore, smart city stores should provide support for:

- *spatial indexing (must have):* providing information near to a given geographical point: as a GPS location. For example, all the services that are currently closed/unavailable to a given point. It should also support advanced geo-spatial functionalities as being able to manage complex geometries (e.g., information along a cycle path, all elements into a given polygonal area).
    - *high* performance on *spatial* querying.
- *full text indexing (must have):* allowing the integration of semantic queries with keyword based searches on text which can be present into the attributes and class elements, as triples. Subjects and objects of triples can contains relevant text area such as descriptions, street names, locations names, etc.
    - *high* performance on *full text* querying.
- *quadruples* (not only triples) to associate dataset metadata with the loaded triples (*must have*). Triples are produced on the basis of data coming from many different sources. Therefore, it is important to track the data source, with metadata and associated licenses. This feature is particularly useful to solve or process licenses during the data usage from clients and via APIs.
- *some kinds of inference* (good to have) such as the basic RDFS or the more advanced OWL2 profiles allowing the inference of new facts from the available data. This may be used to generalize/specialize about entities, to same-as, equivalence, transitive, symmetrical, etc. The inference may imply the materialization of triples in the phase of indexing [Bellini et al., 2015].
- *temporal indexing* (good to have): many information and features are changing over time in smart cities. For example, the weather situation and its related forecast, the traffic flow detected from traffic sensors, the position of buses, and events occurring within the city. For this reason, it is quite important that the RDF store should support temporal search to allow the easy retrieval of temporal data. Moreover, the storing of temporal data (that may change in real time) is the main source for increasing the database size, demanding big data solutions for smart city for volume, velocity and variety, at least.
- *high volume of queries* (good to have). Dealing with bigdata RDF store with many users querying the data is quite challenging, for this reason a clustering solution is needed. It could be a clustering (vertical scale or scale up/down) when the same service is duplicated to allow many concurrent queries and to provide also a fault tolerance solution. It could be also a scale out clustering (horizontal) when data are split among different servers, as a single server cannot handle all the data.

A very relevant non-functional requirement is due to the fact that when it comes to Smart City applications, they are often exploited by Public Administrations. They ask for: (i) standard solution to avoid the risk of vendor lock-in especially for very new technologies like RDF stores are; (ii) open source solutions to be compliant with typical national laws encouraging open solutions with source code accessible and shareable among several public administrations. Moreover, there should be an active community handling and supporting the product.

[2] https://www.w3.org/wiki/RdfStoreBenchmarking

III. Evaluation Methodology

The Smart *City RDF Assessment Model and Benchmark* evaluation methodology is carried out within two phases.

In the **first phase** the *Smart City RDF Assessment Model* is applied. It consists in an analysis of some general features according to the requirements provided in Section II, and more particularly to verify if the RDF/graph store provides support for: SPARQL v.1.1, inference, triples or quadruples, etc.

In the **second phase**, the *Smart City RDF Benchmark* is applied. It is based on performance tests grounded on a set of SPARQL queries designed by considering all the aspects, and including spatial and full text searches (in many cases the SPARQL queries have been designed by adopting the specific constructs related to the different stores). The execution of the Benchmark consists in assessing the performance on the identified queries on three datasets with growing size expanding temporal horizon (1 month, 2 months and 3 months of cumulated real-time data).

A. Smart City RDF Assessment Model

As to the Smart City RDF assessment model, the features considered to analyse the RDF stores are the following:

- *SPARQL version* supported being 1.0 or 1.1;
- inference type supported as full materialization of triples at load time or materialization at query time, and the inference profiles supported (e.g., RDFS, RDFS+, OWL, OWL2, OWL2-DL, …);
- If the store is a triple or quadruple store, check whether it stores only the *subject predicate object* or it can have also a *context* URI;
- How the triples/quadruples are physically stored, namely by using a custom indexing or an RDBMS or other external service (e.g., HBase, Cassandra);
- If the store supports *Clustering* where replicated nodes are used for high availability and fault tolerant solution;
- If the store supports *Scale Out Clustering* where data are allocated on multiple nodes, while no node contains all the data (index sharing);
- If the store supports *Spatial search* at Basic level (meaning that it is able to index and retrieve only geolocated points) or at Advanced level (meaning that it is able to index complex shapes, for example polylines);
- If the store supports full text search, providing the ability to search using keywords;
- If the store allows the association of triple/quadruples with a temporal validity contexts, thus allowing to easily filter triples by means of temporal constraints;
- Size of stores managed as the largest number of triples/quadruples reported to be managed by the RDF store in the literature;
- License under which the RDF store is available, being either open source or commercial;

Table I. Queries of Smart City RDF Benchmark: query name, description and if the query exploit inference or not.

Query	Description	inference	Geo-spat.	Full-text
Find-address	given the latitude and longitude position it retrieves the nearest address within 100m.	No	Yes	No
Municipalities-florence	It retrieves the list of municipalities within the province of Florence.	No	No	No
Bus-lines	It retrieves the list of bus lines.	No	No	No
Bus-stops-of-line	given the bus line, it retrieves the complete bus stop list of the line.	No	No	No
Lines-of-bus-stop	given a bus stop, it retrieves the lines going past that bus stop.	No	No	No
Bus-stop-latlng	given a position and a radius, it finds the bus stops that are within the radius.	No	Yes	No
Bus-stop-florence	It retrieves all the bus stops in the municipality of Florence.	No	No	No
Bus-stop-forecast	given a bus stop, it finds the next forecasts for the lines going past that bus stop.	No	No	No
AVM-distribution	It retrieves for each day the count of the received AVM records.	No	No	No
Service-florence	It retrieves all the services in the municipality of Florence.	Yes	No	No
Service-Acc-Clt-Trs-W&F-florence	It retrieves all the services in the Accommodation, Cultural Activity, TourismService and Wine&Food classes within the municipality of Florence.	Yes	No	No
Service-Htl-B&B-florence	It retrieves all the services in the Hotel and Bed&Breakfast classes within the municipality of Florence.	Yes	No	No
Service-latlng	It retrieves the services within a radius from a latitude, longitude position.	Yes	Yes	No
Service-Acc-Clt-Trs-W&F-latlng	It retrieves all the services in the Accommodation, Cultural Activity, TourismService and Wine&Food classes within a radius from a position.	Yes	Yes	No
Service-Htl-B&B-latlng	It retrieves all the services in the Hotel and Bed&Breakfast classes within a radius from a given position.	Yes	Yes	No
Full-text	It retrieves anything matching a keyword	No	No	Yes
Service-text-florence	It retrieves all the services in the municipality of Florence matching a keyword.	Yes	No	Yes
Service-text-latlng	It retrieves all the services matching a keyword given a position and a radius.	Yes	Yes	Yes
Sensor-florence	It retrieves all the sensors within the municipality of Florence.	No	No	No
Sensor-latlng	It retrieves all the sensors within a radius from a position.	No	Yes	No
Sensor-status	It retrieves the latest information associated with a sensor.	No	No	No
Sensor-distribution	It finds for each day the count of the received sensor status updates.	No	No	No
Parking-status	It retrieves the latest information associated with a parking lot.	No	No	No
Parking-distribution	It retrieves for each day the count of the acquired parking status records.	No	No	No
Weather-florence	It retrieves the latest forecast available for the municipality of Florence.	No	No	No
Weather-distribution	It retrieves for each day the count of the acquired weather forecasts.	No	No	No

- Development language (e.g., Java, C);
- If the project is still an active project, date of last activity, date of last release;

Detailed performance testing should be performed on stores that support minimum set of requirements and in particular providing at least support for:

- SPARQL 1.1 as it provides aggregation functions (group by, count) and other features that were missing in 1.0;
- RDFS inference at load time or query time;
- Quadruples, so that correct metadata can be associated with datasets;
- basic spatial search to allow searching services via position;
- full text search to be able to integrate keyword search with semantic search;
- "Big stores" management in some how: that is the capability of managing large data store with some technique, scaling for instance.

If the RDF store supports additional features, they are positively considered in the context.

B. *Smart City RDF Benchmark*

In this section, the queries at the basis of the *Smart City RDF Benchmark* are presented. The queries performed over the datasets are mainly the ones behind a real Smart City application and the API adopted in Km4City and used in http://servicemap.km4city.org. ServiceMap is an accessible smart city web application for developers to develop informative totems, while the Km4City API is a set of services accessible from Smart City mobile app delivered on all the available platforms: Apple Store, Google Play, and Windows Market.

Noteworthy is that the SPARQL recommendation does not cover the geo-spatial queries, nor the full-text queries. Therefore, in order to support those features, RDF store builders/vendors implemented these features by using their own specific syntax. Due to this reason, for some queries there is not a unique formulation and the query has to be adapted for each RDF store under test (they can be accessed from the web page of the proposed benchmark http://www.disit.org/smartcityrdfbenchmark). In Table I, the semantic queries at the basis of the *Smart City RDF Benchmark* are described and what is highlighted is whether the single query involves in its definition according to the ontology the exploitation of: inference[3], geo-spatial and/or full-text aspects.

For example, the query to retrieve the last weather forecast for the municipality of Florence is the following:

```
PREFIX …
SELECT ?day ?desc ?minTemp ?maxTemp ?time ?wPred
WHERE {
 {
  SELECT DISTINCT ?wRep ?time WHERE {
   ?munic rdf:type km4c:Municipality;
     foaf:name "FIRENZE";
     km4c:hasWeatherReport ?wRep.
     ?wRep km4c:updateTime/schema:value ?time.
  } ORDER BY DESC(?time) LIMIT 1
 }
 ?wRep km4c:hasPrediction ?wPred.
 ?wPred dcterms:description ?desc;
   km4c:day  ?day;
   km4c:hour "giorno"^^xsd:string.
 OPTIONAL { ?wPred km4c:minTemp ?minTemp.}
 OPTIONAL { ?wPred km4c:maxTemp ?maxTemp.}
}
```

[3] https://www.w3.org/TandS/QL/QL98/pp/queryservice.html

It uses a sub-query to find the last report received related to the municipality and from this the prediction associated is selected and the associated information is returned.

A query using full text search and geospatial proximity search (using the syntax of virtuoso) is:

```
PREFIX …
SELECT DISTINCT ?ser ?elong ?elat ?sTypeIta WHERE {
 ?ser ?p ?txt.
 ?txt bif:contains "casa".
 {
  ?ser km4c:hasAccess ?entry.
  ?entry geo:lat ?elat;
    geo:long ?elong;
    geo:geometry ?geo.
  filter(bif:st_intersects(?geo,
     bif:st_point(11.26193046,43.77072194), 0.5))
 }UNION{
  ?ser geo:lat ?elat;
    geo:long ?elong;
    geo:geometry ?geo.
  filter(bif:st_intersects(?geo,
     bif:st_point(11.26193046,43.77072194), 0.5))
 }
 ?ser a ?sType.
 FILTER(?sType!=km4c:RegularService &&
?sType!=km4c:Service)
 ?sType rdfs:label ?sTypeIta.
 FILTER(LANG(?sTypeIta)="it")
}
```

As it occurs with all the RDF benchmarks, the SPARQL queries are specifically tuned for a model. In this case, queries have been designed for the model described in the next section. The complete formalization of the queries, as well as the dataset dumps adopted in the tests reported hereafter, are available at http://www.disit.org/smartcityrdfbenchmark

TABLE II. DATASETS CHARACTERIZATION FOR SMART CITY BENCHMARK.

	1 month		**2 months**		**3 months**	
Type	***Triples***	***%***	***triples***	***%***	***triples***	***%***
AVM	8.4M	19%	18M	33%	28M	43.1%
Parking	413k	0.9%	976k	1.8%	1.4M	2.1%
Sensors	900k	2%	1.7M	3.1%	2.2M	3.3%
Weather forecast	15k	0%	23k	0%	23k	0%
Total dynamic	**9.7M**	**22%**	**21M**	**38%**	**32.5M**	**48.5%**
Road graph	33.5M	75%	33.5M	60.3%	33.5M	50%
Services	681k	1.5%	681k	1.2%	681k	1%
Other static	286k	0.6%	286k	0.5%	286k	0.4%
Total static	**34.5M**	**78%**	**34.5M**	**62%**	**34.5M**	**51.4%**
Total	**44.2M**	**100%**	**55.6M**	**100%**	**67.5M**	**100%**

TABLE III. RDF STORES' FEATURES COMPARISON
where: OS=Open Source, Cm=Commercial, H=Horizontal cluster, V=Vertical cluster

	Features											
RDF Store	***SPARQLv.***	***Inference***	***3/4-uple***	***Spatial search***	***Full-text search***	***Storage***	***Temporal search***	***Size (million/billion triples)***	***License***	***Dev. Language***	***Cluster support***	***Active project***
Virtuoso 7.2.4 OS	1.1	RDFS+	4	Adv	Y	RDBMS	N	50BT	OS	C	N	Y
Virtuoso 7.2.4 Comm	1.1	RDFS+	4	Adv	Y	RDBMS	N	50BT	Cm	C	H	Y
Graph DB SE 7.0.1	1.1	OWL2RL	4	Bas	Y	custom	N	10BT	Cm	Java	H	Y
Stardog 4	1.1	OWL2	4	Adv	Y	custom	N	10BT	Cm	Java	H	Y
Oracle 12c	1.1	RDFS, OWL2	4	Adv	Y	custom	N	1TT	Cm	C/Java	H	Y
Apache Jena-Fuseki	1.1	RDFS OWL-Lite	3	Bas	Y	custom (TDB)	N	1.7BT	OS	Java	N	Y
Apache Jena-Fuseki	1.1	No	4	Bas	Y	custom (TDB)	N	1.7BT	OS	Java	N	Y
Blazegraph 2.1.2	1.1	RDFS+	3	Bas	Y	custom	N	50BT	OS	Java	V&H	Y
Blazegraph 2.1.2	1.1	No	4	Bas	Y	custom	N	50BT	OS	Java	V&H	Y
CumulusRDF	1.1	No	3	No	N	Cassandra 1.2	N	120MT	OS	Java	V	(Y)
Strabon	1.1	No	3	Adv	N	RDBMS	Y	500MT	OS	Java	N	(Y)
4store	1.1	No	4	No	N	custom	N	15BT	OS	C	V	N
h2rdf+	1.0	No	3	No	N	HBase	N	2.7BT	OS	Java	H&V	N

C. *Datasets of the Smart City RDF Benchmark*

The data used for the evaluation are based on the KM4City knowledge base [3]. Some of data are static (or quasi-static) data such as (i) the *road graph* modelling the roads, the public administrations; etc. (ii) the "services" available within the city (e.g., restaurants, hotels, cycle paths, …) and associated with the road graph and organized in an hierarchy; (iii) the bus stops, bus lines of the local transportation, (iv) the road sensors available on the roads. Moreover, the Km4City model provides a number of hierarchies and structures, and huge data with geolocations in which the inferential aspects of SPARQL queries can be profitably tested. Three different datasets have been adopted for the assessment. They share the same 'static' information and only differ for the dynamic part, having one, two or three months of historical dynamic data, respectively. In Table II, the numbers of triples for the different parts of the km4city knowledge base are reported. As you can see, the dynamic parts grow from 22% to 48.5% mostly derived from the AVM (automatic vehicle monitoring, of the ITS) that it is generated out of the data coming for only three bus lines, while the static part is mostly based on structural data like road graph with 34.5M triples, in all the cases.

D. *Real-time data set context description*

Since in a real context the dynamic data change regularly (e.g., weather status, AVM, sensors and parking), the behaviour of the RDF stores should be analysed also under dynamic conditions like queries, while other processes are performing update/upload. Moreover, in order to test a more realistic case the queries retrieving the last value of dynamic data (e.g., sensor last value) could be arranged by using a model including triples stating which is the latest obtained value . In this case, a SPARQL query should be used to remove the association with the latest received value and insert the new triple associated with the new reading of values.

To analyse performance on dynamic update conditions a specific test case has been set up (e.g., traffic, IOT). In order to establish replicable conditions, a tool has been used to regularly generate the status of the 430 sensors using the NTriples format (stored in a specific context) as standard SPARQL 1.1 Graph Store HTTP Protocol. They are produced and singularly loaded into the store, together with their association with the latest value to the corresponding sensor. Each submission stores 19 triples for each sensor and thus 8056 new triples are stored about every 30 seconds. In this case, the 3 months dataset of Table II has been used. Together with the process of upload/update, the server runs at the same time all the queries of the benchmark to assess if updating the triples while querying, either influences or not the query time.

TABLE IV. RDF STORES PERFORMANCE OF DATA LOADING,
"NA" MEANS THAT THE INFORMATION IS NOT AVAILABLE (IMPOSSIBLE TO MEASURE)

	Triples load time	**Stated triples**	**Stored triples**	**Size (of which: full text index size, spatial index size)**
GraphDB – 1 month	1h 8m	44,274,756	84,425,185	8.5GB (299MB, 66MB)
GraphDB – 2 months	1h 48m	55,617,333	104,041,312	10GB (379MB, 67MB)
GraphDB – 3 months	2h 10m	67,082,202	124,015,329	13GB (459MB, 70MB)
Virtuoso – 1 month	16m	44,274,820	46,259,439	2.2GB (NA, NA)
Virtuoso – 2 months	21m	55,619,789	57,669,629	2.8GB (NA, NA)
Virtuoso – 3 months	31m	67,084,661	69,200,459	3.5GB (NA,NA)
Stardog – 1 month	1h 19m	44,273,368	44,273,368	4.8GB (341MB, 131MB)
Stardog – 2 months	1h 24m	55,615,945	55,615,945	5GB (318MB, 129MB)
Stardog - 3 months	2h 58m	67,080,814	67,080,814	6.2GB (493MB, 138MB)
Oracle – 1 month	6h 18m	44,270,460	78,744,647	25GB (NA, NA)

IV. COMPARING RDF STORES WITH SMART CITY RDF ASSESSMENT MODEL

In this section, the RDF stores under assessment are compared according to the feature model which has been identified and discussed in Section III.A. The comparison is carried out with the aim of identifying the stores that are better ranked to be used on smart city applications in terms of provided features.

In Table III, the features supported by the different RDF stores under evaluation are summarized and the values considered as minimum requirements are highlighted. A description of the RDF stores considered in the assessment and reported in Table III is given below.

Virtuoso 7.2.4 [18], it is mostly known because it is the RDF query engine behind dbpedia.org. It is a SPARQL 1.1 quadruple store developed in C available both via open source and commercial license. The open source version mainly misses the clustering feature. Inference is not materialized at load/indexing time, while query rewrite is performed to support RDFS+ inference. It is backed by the Virtuoso RDBMS and thus SPARQL queries are translated to SQL for that RDBMS. It supports advanced spatial indexing and supports full text search. The community behind virtuoso is headed by OpenLink Software ltd and it is quite active.

GraphDB SE 7.0.1 (former OWLIM store) [4] is a commercial solution providing a SPARQL 1.1 endpoint supporting triple/quadruple stores with spatial indexing of geographic coordinates and full text indexing based on Lucene, Apache. It supports inference at load/indexing time with different rule sets (RDFS, OWL2RL, etc.), and such rule sets can be selected by the user. It has been told to support up to 10 billion of triples on a single node. The Enterprise edition allows horizontal scaling with a master node forwarding the insert/update/delete operations to slave nodes. The solution is implemented in Java using OpenRDF Sesame. The project is still active and it is managed by Ontotext.

[4] http://ontotext.com/products/ontotext-graphdb/

Blazegraph (ex BigData) [5] is an open source project, providing also a commercial license. It supports triple and quadruple stores. With RDFS+ inference (at load time) it is available only on triple stores. It has a full-text indexing support, and there is a basic geospatial indexing, too. It provides both a horizontal and vertical scaling solution, thus allowing an index to be shared on multiple nodes. A single computer can manage up to 50 billion triples. The project is managed by Systap and it is still active.

CumulusRDF [19] is an open source project based on OpenRDF Sesame using Apache Cassandra 1.2 as NoSQL storage layer. It does not support inference and can store only triples. Since it is based on Cassandra, it supports vertical scaling for storage of the indexes on the nodes in the cluster, while only one node is used to perform queries.

Stardog 4.1.1 [6] is a commercial RDF quadruple store developed by Clark&Parsia (developer of the well-known OWL reasoner Pellet). It supports SPARQL 1.1 and OWL2 inference at query time, full-text indexing and search, and spatial indexing and search. It allows horizontal scaling, and it is a quite active project. Stardog may support 10 billion triples store on single node while the community version manages up to 25 million triples.

Strabon [20] is an open source SPARQL 1.1 store developed to support both spatial and temporal search [Bereta et al., 2013] . It is based on PostGIS extension of Postgres RDBMS; it does not support inference, nor full-text search. It only provides support for storing triples (the context URI associated with the triple is used for temporal linking). No clustering solution is available.

4store[7] is an open source quad RDF store developed in C supporting a clustering solution which stores the quads on different nodes (max 32). It does not support any inference, any full-text search, nor geospatial search. The activity seems to be moved to 5store, which is a corresponding commercial version.

[5] https://wiki.blazegraph.com
[6] http://stardog.com/
[7] http://4store.org/

TABLE V. RDF STORES PERFORMANCE OF NON-SPATIAL QUERIES (BEST PERFORMANCES IN BOLD)

Query	GraphDB			Virtuoso			StarDog			Oracle	Number of results
	1 month (ms)	*2 months (ms)*	*3 months (ms)*	*1 month (ms)*	*2 months (ms)*	*3 months (ms)*	*1 month (ms)*	*2 months (ms)*	*3 months (ms)*	*1 month (ms)*	
Municipalities-florence	7	10	121	8	15	9	127	173	129	2,391	46
Bus-lines	17	18	91	**6**	**7**	**6**	125	156	141	2,325	85
Bus-stops-of-line	**50**	**26**	**28**	65	68	62	194	211	172	36661	135 (max)
Lines-of-bus-stop	**7**	**12**	**18**	21	23	20	210	235	210	6457	11 (max)
Bus-stop-florence	**100**	**113**	**126**	374	291	281	216	258	201	34071	1108
Bus-stop-forecast	**96**	**413**	**444**	632	2065	2008	2028	3072	5084	259577	15
AVM-distribution	914	1893	2767	**26**	**58**	**70**	1442	2417	3772	26844	89 (max)
Service-florence	7106	7841	10150	**2170**	**2135**	**2158**	3689	3667	3514	>10min	3259
Service-Acc-Clt-Trs-W&F-florence	8,158	8274	8318	**2386**	**2917**	**2930**	4118	4110	6416	>10min	1179
Service-Htl-B&B-florence	3311	3296	4,035	**537**	**845**	**766**	3640	3782	3448	>10min	234
Full-text	314	750	618	**64**	**96**	**67**	166202	214344	215937	136243	1389 (max)
Service-text-florence	286842	295057	284573	**1981**	**3621**	**5661**	165860	202919	209364	126833	51 (max)
Sensor-florence	**21**	**48**	**46**	82	93	84	785	615	483	7349	62
Sensor-status	598	1101	1560	**56**	**146**	**163**	295	384	392	173612	1
Sensor-distribution	939	1867	2665	**174**	**328**	**341**	672	1060	1,346	178272	78 (max)
Parking-status	83	188	309	**72**	**87**	**100**	1,388	1339	1053	40823	1
Parking-distribution	455	1096	1628	**61**	**131**	**203**	223	373	451	30444	83 (max)
Weather-florence	**9**	**19**	93	46	60	**71**	181	182	149	5047	5
Weather-distribution	12	23	19	**7**	**18**	**11**	126	141	128	2342	38 (max)

h2rdf+ [21] is an open source triple store based on HBase and Hadoop platform. It supports only the SPARQL 1.0 specification, and does not support any inference, any full-text indexing, nor geo-spatial search. Being based on HBase and Hadoop, it provides horizontal and vertical scaling.

Apache Jena-Fuseki 2.3.1 [8]is an open source SPARQL 1.1 engine integrated within the java based Apache Jena framework. Jena provides the quads RDF storage layer which could be native on file system (TDB), based on a SQL DBMS (SDB) or in memory. Jena provides also the inference support (supporting RDFS, OWL-Lite or using custom rules) but it works only on triple stores and not on quadruples stores, moreover it supports full-text and basic spatial indexing based on Lucene or Solr. No clustering solution has been reported.

Oracle Database 12c, the well-known Oracle database solution provides support for RDF graphs, full-text & spatial indexing/search but it does not support the standard SPARQL HTTP query protocol, it can be integrated by using the open source Jena framework with Fuseki or Joseki tools. Moreover Oracle solution provides inference (RDFS, OWL2RL and custom rules).

As a conclusion of this section, it is self-evident from Table III, that the RDF store solutions supporting all the minimum requirements are Virtuoso 7.2.4 open source and commercial edition, GraphDB Standard Edition 7.2, Stardog 4 and Oracle 12c. Therefore, only these RDF stores have been assessed in term of performance, as reported in Section V.

V. ASSESSING RDF STORES WITH SMART CITY RDF BENCHMARK

The performance evaluation has been carried out by considering: (i) the loading/indexing time for knowledge base initialization, (ii) the execution time without any update for spatial and non-spatial queries, and (iii) query execution time while the sensors data were regularly updated. The performance has been evaluated using a server Ubuntu 14.04 with 8GB RAM, CPU, Intel Xeon E5-2680@2.8GHz with 20 logical processors, HD at 15.000 RPM. Table IV reports the results for the loading/indexing time concerning the different previously discussed datasets, respectively. It should remarked that Virtuoso is the fastest, GraphDB and Stardog perform

[8] https://jena.apache.org

TABLE VI. RDF STORES PERFORMANCE OF SPATIAL QUERIES (THE BEST PERFORMANCES IN BOLD)

	GraphDB			Virtuoso (intersect)			Virtuoso (distance)			Stardog			
Query	1 month (ms)	2 months (ms)	3months(ms)	1 month(ms)	2 months(ms)	3months(ms)	1 month(ms)	2 months(ms)	3months(ms)	1 month(ms)	2 months(ms)	3months(ms)	Number of **results**
Find-address	**47**	**180**	**218**	219	160	143	5762	5965	5776	2495	2848	2367	1
Bus-stop-latlng(100m)	**8**	**8**	**9**	33	63	63	28	31	28	2105	1791	1885	1 (max)
Bus-stop-latlng(500m)	33	52	48	147	182	166	**34**	**41**	**33**	1781	1853	1810	20
Bus-stop-latlng(1km)	82	135	155	76	--	265	**42**	**43**	**47**	2095	2116	2334	93 (max)
Bus-stop-latlng(5km)	463	669	782	--	--	--	**201**	**201**	**205**	2883	3245	2815	1003 (max)
Service-latlng(100m)	691	1111	324	2788	2117	915	**582**	**761**	**754**	3970	4581	5123	41 (max)
Service-latlng(500m)	1131	1256	1212	627	513	549	**401**	**421**	**391**	4110	4303	4467	784 (max)
Service-latlng(2km)	6087	6550	6548	1377	1688	--	**1062**	**1167**	**1079**	5832	6929	6204	3718 (max)
Service-latlng(5km)	11192	13069	12712	3054	--	--	**1900**	**1996**	**1912**	6585	7494	6551	6666 (max)
Service-Acc-Clt-Trs-W&F-latlng(100m)	**74**	**125**	**87**	880	921	773	1260	1209	1073	5978	6076	4986	37 (max)
Service-Acc-Clt-Trs-W&F-latlng(500m)	**948**	**1091**	**1602**	2159	1709	1698	1159	1187	1232	6130	6738	5933	650 (max)
Service-Acc-Clt-Trs-W&F-latlng(2km)	4731	5644	7999	--	--	--	**1706**	**1807**	**1619**	6451	8669	7353	2224 (max)
Service-Acc-Clt-Trs-W&F-latlng(5km)	7983	9610	9974	--	--	--	**1785**	**1938**	**1813**	8024	9669	7646	3102 (max)
Service-Htl-B&B-latlng(100m)	**29**	**38**	**29**	420	466	392	541	563	631	3843	3985	3886	7 (max)
Service-Htl-B&B-latlng(500m)	**293**	**371**	**393**	1119	724	1032	555	666	544	4055	4605	4424	151 (max)
Service-Htl-B&B-latlng(2km)	1390	1729	2020	--	--	--	**617**	**683**	**681**	5664	6573	6278	488
Service-Htl-B&B-latlng(5km)	2421	3144	8281	--	--	--	**673**	**744**	**682**	6516	7009	7053	611(max)
Service-text-latlng(500m)	204936	236937	256328	433	148	324	**242**	**64**	**73**	>7min	>7min	>7min	21 (max)
Sensor-latlng(100m)	**10**	**12**	**13**	--	--	63	20	27	30	1842	1639	1604	0
Sensor-latlng(500m)	41	62	198	118	73	163	**18**	**23**	**26**	1788	2069	1923	4
Sensor-latlng(2km)	229	335	444	--	--	--	**22**	**29**	**29**	2372	2798	2670	29
Sensor-latlng(5km)	514	721	888	--	--	--	**23**	**30**	**38**	2961	2947	2837	56

similarly (about 5 times slower than Virtuoso) and Oracle is the slowest being about twenty three times slower than Virtuoso. Due to the performance of Oracle 12c in loading, the decision was to test only the 1 month data set case. On the other hand, GraphDB and Oracle perform inference at load time while Virtuoso and Stardog at query time, under user request. For this reason, the number of triples indexed by GraphDB is typically 80% bigger than those of Virtuoso. As to Virtuoso, the slight increment of triples stored/indexed with respect to the ones provided to the RDF store (2.1M for the 3 months case) is due the transformation of the geo:lat and geo:long triples in a geo:geometry with POINT() to enable the geo-spatial indexing. While in the same case, as to GraphDB, the increment of about 57M of triples is due to the materialization of triples via inference at the indexing/loading time.

Tables V and VI focus on the results for the query execution time concerning GraphDB, Virtuoso, Stardog and Oracle and related to the different time horizons of one, two and three months, respectively. Table V reports the performances for non-spatial queries and Table VI for spatial queries. The queries have been tested performing a pseudo-random sequence of 1000 queries repeated two times with some pseudo-random arguments in order to reduce the caching effect. The sequence of performed queries has been the same for each test execution, so as to test the same sequence on different systems. The table reports the maximum number of results obtained for each type of query, when the number of results depends on the parameter randomly chosen (e.g., lines of a bus stop) or from the different dataset used (e.g., the AVM, sensor, parking and weather distribution queries). When considering the poor performance by Oracle 12c in loading and also in the query times, it was decided to test only the 1 month case. Moreover, a bug in the Oracle plugin for Apache Jena did not allow to perform spatial queries via the HTTP protocol and this is the reason why Oracle 12c does not appear in Table VI.

If observing the query results (see Table V), when no spatial and full text search and inference are involved, the performances of Virtuoso and GraphDB are comparable, in some cases GraphDB is even better ranked. When inference is needed (e.g., in the test cases *Service-florence*, *Service-Acc-Clt-Trs-W&F-florence, Service-Htl-B&B-florence*), as to Virtuoso the inference had to be enabled on the single constraint involving a general class (e.g., all services in the Accommodation class). While if the inference is enabled, generally on the query, the internal automated query rewrite takes a longer time (may be related to the size of the exploited ontology). For example, for query *Service-Acc-Clt-Trs-W&F-florence* the time grows from an average of 2.9s to an average of 19.6s (on the 3 months dataset). In those cases, the GraphDB results are better ranked. Stardog generally is the slowest on all the queries.

When considering the spatial indexing (see Table VI) in Virtuoso, some mistakes have been detected using the *st_intersection* function. In some cases, Virtuoso returned an error, in other cases it provided a smaller number of results than the correct number; Virtuoso could provide different results for the same query for the three different datasets, even if they do not differ for the part considered in the query. On the other hand, in Virtuoso, if the *st_distance* function is used, all the obtained results have been verified to be correct, apart from few cases on the border (due to the numerical computation in measuring distances). The usage of the distance function for Virtuoso is a good solution in most cases, while the query optimizer seems to avoid the exploitation of the spatial index. This fact may be deduced out of a comparison among the results of the formalization of query *Find-address:* in two cases by using: (i) *st_distance* function it takes about 5.7s, while (ii) with *st_intersect* function it takes about 0.14s.

TABLE VII. SENSOR DATA UPLOAD PERFORMANCE

	GraphDB	Virtuoso	Stardog
total mean time (ms)	4135.59	1290.05	42498.80
mean upload time (ms)	2105.06	893.52	41526.02
mean update time (ms)	2030.53	396.53	972.78
minimum total time (ms)	1810.00	480.00	6050.00
maximum total time (ms)	37294.00	20678.00	791083.00
std. dev of total time (ms)	2197.81	2082.30	76121.02

TABLE VIII. STORE PERFORMANCE IN PRESENCE AND ABSENCE OF UPDATES DURING BENCHMARK

RDF store	MNQPH		
	no updates	during updates	Loss in performance
GraphDB	2117.00	1799.93	14.97%
Virtuoso	4584.16	4362.21	4.84%
Stardog	1620.24	876.63	45.89%

TABLE IX. PERFORMANCE IN ACCESSING TO THE LAST VALUE OF SENSORS

RDF store	Mean Time to Sensor Status access, Time in ms		
	Case 1 (no update)	Case 2 (no update)	Case 3 (update)
GraphDB	1561	31	334
Virtuoso	163	46	174
Stardog	393	208	554

Another aspect to be considered is the mixing of spatial query with text search query (for example, in query *Service-text-latlng(500m)*). With GraphDB and also with Stardog, we obtained a higher execution time, hitting in some cases the timeout. In this case where spatial and text are combined for Virtuoso, the intersect function returned an error, while the distance function performed very well.

Regarding the analytic queries (for example: *Weather-distribution, AVM-distribution*) which count the daily number of records of the weather forecasts, bus, sensor data, parking status for the three datasets, both solutions have provided acceptable execution time (less than 5s). In this case, Virtuoso is better ranked with less than 0.5s of execution time. Moreover, Virtuoso presents a less growing factor with respect to GraphDB.

A. Assessing query execution time under update/load

During the test, the time to upload/update new triples for all the sensors and mark them as the '*latest value*' has been recorded and reported in Table VII. Therefore, the minimum, maximum, average time and the standard deviation of the upload and update time are reported for each RDF store. From the results, it is clear that Virtuoso turned out to be the smartest, since it performed the update of the 430 sensors within 20s, while GraphDB did the same in 37s, and StarDog had an average of 42s and with a maximum time in just one case of 13 minutes to upload new triples for all the 430 sensors. As to Oracle with Apache Jena-Fuseki it was not possible to send the triples for the 430 sensors through Fuseki, since the communication was hanging; while when sending the data for only 10 sensors the average time was about 16s with a maximum of 2.5 min.

In order to evaluate the impact of the update/upload action on query performance, the mean number of query per hour (MNQPH) has been computed for each RDF store in presence or absence of ongoing upload/updates. MNQPH has been computed as the ratio of total time needed to run a large number of queries of the benchmark and the number of queries. In particular, some of the queries such as: "Service-text-latlng(500m)", "Service-text-florence" and "Full-text" have been removed because they typically generate on GraphDB and StarDog many timeouts, that could create too noise on assessing query performance during update/load activity.

The results are reported in Table VIII, where you can see that the MNQPH is decreased in all the cases, shifting from the value registered with RDF store under no updates up to the value registered during the store updates. The decrease in performance is due to the fact that the query has to wait for the store unlock. Among the RDF stores considered, Virtuoso presented the lower reduction in performance. Moreover, as stated above, the benchmark occurred in some time outs with to GraphDB and StarDog stores in the absence of updates; typically 46 and 96 times for the whole benchmark. The number of timeouts is more than twice in presence of updates.

Table IX reports the mean time to get access to the latest value of a sensor series (the *Sensor-status* query) in three cases: (1) using the order by clause and without concurrent updates, (2) using the "latest value" triple without concurrent updates and (3) using the latest value triple during concurrent updates. For all the stores we can see that when avoiding the sort and using the "latest value", the time needed to access is reduced. However, performing a concurrent update increases access time of a significant amount (i.e., more than 10 times for GraphDB, 3.8 times for Virtuoso and 2.7 for StarDog). According to the access mean time values, Virtuoso could perform better than the others in all the cases.

VI. Conclusions

The usage of RDF stores to store smart city data is becoming of wide interest for several applications. In this paper we have proposed a *Smart City RDF Assessment Model* for a comparative study about the state of the art on RDF stores according to their main features and in particular on the SPARQL aspects/features. In addition, the *Smart City RDF Benchmark* has been proposed. The benchmark is based on (i) some datasets of triples (that are grounded on Km4City ontological model) accessible from http://www.disit.org/smartcityrdfbenchmark, it can be used only for benchmarking purpose; (ii) a set of SPARQL queries declined for different SPARQL constructs. The benchmark has been defined for smart city services to compare results which can be obtained by using different RDF Stores.

The comparison addressed a number of well-known RDF stores such as Virtuoso, GraphDB, StarDog, and Oracle for the performance aspects. As a general consideration about performance, it should be noted that Virtuoso performs better in presence of less selective queries, thus providing a higher number of results. On the contrary, GraphDB performs better when specific results are searched, thus when a smaller number of results are requested.

Acknowledgement

The authors would like to say thanks to ONTOTEXT for granting us the access to a trial version of their RDF store. This works has been developed in the framework of Km4City activity for RESOLUTE H2020, and for REPLICATE H2020 (both are European Commission funded International Projects) and for Sii-Mobility Smart City National project (http://www.sii-mobility.org).

References

[1] Mulligan, C.E.A.; Olsson, M., "Architectural implications of smart city business models: an evolutionary perspective," Com. Magazine, IEEE, vol.51, no.6, pp.80,85, June 2013

[2] Welington M. da Silva, Alexandre Alvaro, Gustavo H. R. P. Tomas, Ricardo A. Afonso, Kelvin L. Dias, and Vinicius C. Garcia. 2013. Smart cities software architectures: a survey. In Proc. of the 28th Annual ACM Symp.on Applied Computing (SAC '13). ACM, New York, NY, USA, 1722-1727.

[3] P. Bellini, M. Benigni, R. Billero, P. Nesi, N. Rauch, "Km4City Ontology Bulding vs Data Harvesting and Cleaning for Smart-city Services", International Journal of Visual Language and Computing, Elsevier, 2014

[4] Anthopoulos, L.; Fitsilis, P., "Exploring architectural and organizational features in smart cities," in Advanced Com. Technology (ICACT), 16th Int. Conf. on, vol., no., pp.190-195, 16-19 Feb. 2014.

[5] Zaheer Khan, Ashiq Anjum, and Saad Liaquat Kiani. 2013. Cloud Based Big Data Analytics for Smart Future Cities. Proc. of the IEEE/ACM 6th Int.Conf. on Utility and Cloud Computing (UCC '13). Washington, 381-386.

[6] Zaheer Khan, Ashiq Anjum, and Saad Liaquat Kiani. 2013. Cloud Based Big Data Analytics for Smart Future Cities. In Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC '13). IEEE Computer Society, Washington, DC, USA, 381-386.

[7] Y. Guo, Z. Pan, and J. Heflin. "Lubm: A benchmark for owl knowledge base systems". J. Web Semantics, 3(2-3):158–182, 2005.

[8] C. Bizer, A. Schultz. "The Berlin SPARQL Benchmark". International Journal on Semantic Web & Information Systems, Vol. 5, Issue 2, Pages 1-24, 2009

[9] G. Garbis, K. Kyzirakos, M. Koubarakis. "Geographica: A Benchmark for Geospatial RDF Stores". In the 12th Int. Semantic Web Conf. (ISWC). Australia, Oct. 21-25, 2013

[10] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. "Apples and oranges: a comparison of RDF benchmarks and real RDF datasets". In Proc. of the 2011 ACM SIGMOD Int. Conf. on Manag. of data (SIGMOD '11). ACM, New York, NY, USA, 145-156, 2011

[11] Schmidt, Michael, et al. "SP^ 2Bench: a SPARQL performance benchmark." Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009.

[12] D. F. Barbieri, D. Braga, S. Ceri, E. Della Valle, and M. Grossniklaus. "C-sparql: Sparql for continuous querying". In Proc. of WWW, pages 1061–1062. ACM, 2009.

[13] J.-P. Calbimonte, O. Corcho, and A. J. G. Gray. "Enabling ontology-based access to streaming data sources" In ISWC, pp. 96–111, 2010

[14] Y. Zhang, M.-D. Pham, O. Corcho and J.-P. Calbimonte. "SRBench: A Streaming RDF/SPARQL Benchmark" In Proc. of the 11th International Semantic Web Conference ISWC 2012. Boston, USA, Nov 2012.

[15] Le-Phuoc, D., Dao-Tran, M., Pham, M. "Linked Stream Data Processing Engines: Facts and Figures" In International Semantic Web Conference (ISWC 2012). Volume 1380, Boston, USA, Springer (2012) 300–312

[16] Muhammad Intizar Ali, Feng Gao, Alessandra Mileo, "CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets", 14th Int. Semantic Web Conference, Bethlehem, PA, USA, October 11-15, LNCS, 2015

[17] P. Bellini, I. Bruno, P. Nesi, N. Rauch, "Graph Databases Methodology and Tool Supporting Index/Store Versioning", publication on JVLC, Journal of Visual Languages and Computing, Elsevier, 2015.

[18] O. Erling and I. Mikhailov. "Virtuoso: RDF Support in a Native RDBMS". In Semantic Web Information Management, pages 501-519. Springer, 2009.

[19] G. Ladwig, A. Harth, "CumulusRDF: Linked data management on nested key-value stores", 7th International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2011), 2011.

[20] K. Kyzirakos, M. Karpathiotakis and M. Koubarakis. "Strabon: A Semantic Geospatial DBMS". In the 11th International Semantic Web Conference (ISWC 2012), Boston, USA, 11-15 November 2012.

[21] N. Papailiou, I. Konstantinou, D. Tsoumakos, P. Karras, N. Koziris, "H2RDF+: High-performance distributed joins over large-scale RDF graphs", Big Data, 2013 IEEE International Conference on , pp.255,263, 6-9 Oct. 2013

Context Awareness for e-Tourism: an Adaptive Mobile Application

Francesco Colace
DIIn
University of Salerno
Via Giovanni Paolo II, 132, Fisciano (SA), Italy
fcolace@unisa.it

Saverio Lemma, Marco Lombardi
SIMASLab
University of Salerno
Via Giovanni Paolo II, 132, Fisciano (SA), Italy
{slemma, malombardi}@unisa.it

***Abstract*—The Italian towns have a cultural heritage that often do not succeed in being completely enhanced. The natural, artistic and cultural resources present in the Italian towns, above all the smallest ones, many times remain hidden and are not enjoyed by the tourists. In this paper, it is introduced an Adaptive Context Aware app able to support a tourist inside a town. The system can guide the tourist in the discovery of a town proposing him/her resources and services mainly interesting for the user according to his/her interests and the position where he/she is. The objective is reached through the use of a system of description of the context through a graphical formalism named Context Dimension Tree. The App collects information also from social environments adapting the proposed itinerary taking into account the communities and the interests of the user. The entire approach has been tested inside the town of Salerno with very interesting results.**

Keywords-Context Awareress, Mobile Application, Pervasive Computing.

I. Introduction

The Italian towns have a cultural heritage that often do not succeed in being completely enhanced. The natural, artistic and cultural resources present in the Italian towns, above all the smallest ones, many times remain hidden and are not enjoyed by the tourists. This problem typology becomes even more important when the tourist has few hours to visit a town: think, for instance, about some passengers of a cruise who in few hours have to visit an unknown place. The problem arises also for those people who, for work, live an experience in a town that they can visit in little time. Where eating? What seeing? How moving? These are the typical questions that such a user makes when he/she is in a station, an airport or a harbor. If in the big towns there are pre-constituted itineraries that can be easily made by the tourists, this is not always true in towns of little or medium dimension that, even if they have a sure interesting cultural heritage, often risk of not enhancing it completely.

On second thoughts, information necessary for the enhancement of the resources of a town are, in many cases, already present on the web: the social networks have much information about the resources present in a town. On the other hand, also the public institutions, usually, develop some contents in support of the cultural resources present in the territory, but not present in places not easily reachable by the tourists, above all the foreign ones. Moreover, often, there are also services that can be useful for a tourist who unlikely knows where finding them. Therefore, it is necessary to create a framework that can integrate contents and services to support a user inside a certain territorial context.

The adoption of Future Internet (FI) technology and of its most challenging components like the Internet of Things (IoT) and the Internet of Services (IoS), can constitute the basic building blocks to progress towards a unified ICT platform for a variety of applications within the large framework of smart cities projects [1]. In addition, recent issues on participatory sensing, where every day mobile devices like cellular phones form interactive, participatory sensor networks enabling public and professional users to gather, analyze and share local knowledge [2], seem to fit the smartness requirements of a city in which also people have to play an active role. Eventually, the cloud computing technologies provides a natural infrastructure to support smart services [7].

As previously said, one of the fields that can take great advantages from such technologies is Tourism [3]. In this scenario, persons (citizens, tourists, etc.) and objects (cars, buildings, rooms, sculptures, etc.) equipped with appropriate devices (GPS, smart-phone, video cameras, temperature/humidity sensors, etc.) constitute a particular social network in which all the mentioned entities can communicate [4].

Exchanged and produced data can be exploited by a set of applications in order to make the system "smart". From a more general point of view, the social network can be seen as composed of a set of Single Smart Spaces (S3) (indoor museums, archaeological sites, old town centers, etc.), each needing particular ICT infrastructure and service that transforms the physical spaces into useful smart environments. Here, one of the most challenging and interesting research problem is to model context-awareness in a S3 and design context aware applications able to provide useful data and services depending on the current context occurrences [8][9].

Context is not just a simple profile that describes the surroundings of data. Rather, context is better described as any piece of information that can be used to characterize the situation of an entity such as a person, a place, or any other relevant object/aspect in the interaction between a user and an application. In

DOI reference number: 10.18293/DMSVLSS2017-010

this paper, we try to give an answer to the problem of the context representation using the Context Dimension Tree formalism [5][6][10].

On the basis of what has been previously described, this work will be organized in this way: in the following paragraph, we will describe the concept of context and how it can be declined in a modern way thanks to the use of new technologies. Then, we will introduce a context-based approach able to give, inside a little town, services and contents useful for the user. Some experimental results will be presented in the last part of the paper.

II. Motivating Example

In this section, we describe a typical application in the Tourist domain in order to better understand the main features of the proposed system. In particular, we consider a tourist that during her/his vacation in Campania desires to visit Salerno, a beautiful town located in the South of Italy. To be considered smart, the environment related to the town of Salerno should provide a set of smart services for:

a. suggesting the visit of the most important cultural places in Salerno
b. having information about the restaurants in Salerno
c. accessing to proper multimedia guides describing the main artworks that are in Salerno
d. recommending special visit paths (trekking paths, bicycle tours, …)
e. monitoring the weather condition
f. showing the timetable of the transport services located in Salerno
g. saving the visit in a multimedia album
h. accessing to information about public services in Salerno (Post Office, Pharmacy, …)

For improving their effectiveness, these services and contents have to be furnished to the user in the right context and at the right timing. Therefore, it is important the context awareness of the framework and the opportunity to use it by mobile devices [11]. Another important feature of the system is the ability to suggest resources that usually are not considered as mainstream.

In order to give the most suitable contents to the users, in this paper we introduce a context aware system able to tailor data and services depending on the context and the users' needs. For example, if the user declared as preference the use of the transportation when he/she is close to a bus stop the timetable will be automatically downloaded on his/her smartphone. The same will happen when the user is close to restaurants: if he/she loves food based on fish, only this kind of restaurant or menu will be proposed. Data about resources and services are collected from a knowledge base built by a group of experts and collecting information from the various social networks.

In the next paragraphs, more details about the system architecture and the application of the proposed approach in real context will be furnished.

III. Context Awareness and ICT

The human being has always used the concept of context, which belongs to that kind of concepts known by the majority of people, but that are difficult to describe with words. In [17], there is the first attempt to describe the relation between the context and the context-awareness in the field of information technologies. The three main aspects of the context are: 'where you are', 'who you are with' and 'what resources are nearby'. If we put together the three just mentioned sentences, we realize that they can be seen as a first definition of context based on some observable characteristics. Another definition of context has been proposed in [16] where the context is defined as a series of environmental features (environment), such as, for example, the place, the temperature and the considered user's identity. The definition of context that usually is taken into account is that proposed by [15]: 'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.' Directly linked to the definition of context, there is that of context-awareness applications: applications that in some way are aware of the context where the user is and the capability to detect and react to the changes in the environment where it is located [14]. Again in [15], there is a definition of the context-aware system: 'A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.' In practice, a system can be defined context aware when it takes advantage of the context to give important information and/or services to the user, where the importance depends on the user's request and features. If we wanted to classify the context-aware applications, we could consider that presented in [17]:

1. Proximate Selection, which literally means 'selection of proximity', is an interfacing technique that considers that the user gets close to a particular place to receive some relevant information and/or to make elaborations, both on request and automatically.
2. Automatic contextual reconfiguration is the process of addition of new components, removal of already existing components or alteration of the connection among the components of a system. In actual fact, there is the change of the system according to the context. Typically, the components can include: driver modules directly downloadable by the user, modules of programs.
3. Contextual information and commands: often the operations that people make can be predicted. In fact, usually, there are some recurring operations made in particular places (e.g. universities, libraries, offices, etc.). The applications that use this kind of 'contextual information' are made to accomplish certain orders (contextual commands) in place of the user according to the context.
4. Context-triggered actions are those applications that automatically carry out an operation when there is a particular condition (trigger) in the context.

Although as time goes by new classifications have been introduced, the previous ones are still valid. It is important to precise that a context-aware application has not to necessarily belong to one of the listed categories, but it is possible to have some 'hybrid' applications that have features belonging to more categories.

What are the context-aware applications for? For some years, more and more often we hear talking about smart environments aimed to the improvement of the quality of life, both in domestic environment (domotics) and in city environment. In particular, there is an expression that recurs a lot in the several mass media: 'smart city'.

The smart cities are the so-called intelligent cities. This subject is interdisciplinary and encompasses all the fields: from the energy saving, to the improvement of life and the fastest and more natural access to information. It is exactly in these two latest fields that the context- aware applications insert themselves. In fact, in the future cities, there will be more and more smart spaces (domestic and not), which will take care of the users making easier and more immediate their access to information and, under determined conditions, will be able to foresee the user's desires and therefore to anticipate some operations on behalf of the user. As an example of smart environment, we can think of a room that has the capability of automatically regulating the temperature of the environment according to the user's preferences, or, through a centralized stereo system, can change music according to the user's tastes. And moreover, we could think to a public park where people, by tagging, can leave their own messages on a virtual wall, so that in the future the users of this same park can take advantage of the advices of who has been previously in that place.

A further example could be that of a smart shopping center, where when a user enters in a shop directly receive information about the products on sale that he/she could be interested in. This processing can be made on the basis of the previous purchases and/or of a series of indications given by the same user (for example, through an electronic questionnaire made available by the shopping center).

This kind of applications can become very important also in the field of the improvement of disabled people's life. In fact, it is possible to study some areas that change according to the specific need. For example, let us consider a blind person that enters a smart public building, this environment, after having received context information about the user, has to be able to guide him/her towards his/her destination using audio messages.

In the following paragraph, we will present an approach to the management of the context and its associated services to make the previously introduced approachesconcrete.

IV. A Context Dimension Tree Based Approach

A key element in the design of a contextual application and a Context Aware System is the representation and management of the context itself. To better understand formal concepts, it has been carried out in the paper an example based on a simplified citizen domain, on which it is now being developed a Context Aware System that assists residents and tourists in their activities.

The goal is to provide a mechanism of dynamic and automatic invocation of services considering the context through the Context Dimension Tree [18].

CDT is a tree composed of a triad <r; N; A> where r indicates its root, N is the set of nodes of which it is made of and A is the set of arcs joining these nodes.

CDT is used to be able to represent, in a graphic form, all possible contexts that you may have within an application.

Nodes present within CDT are divided into two categories, namely dimension nodes and concept nodes. A dimension node, which is graphically represented by the color black, is a node that describes a possible dimension of the application domain; a concept node, on the other hand, is depicted by the color white and represents one of the possible values that a dimension may assume. Each node is identified through its type and a label.
The children of the root node r are all dimension nodes, they are called top dimension and for each of them there may be a sub-tree. Leaf nodes, instead, must be concept nodes. A dimension node can have, as children, only concept nodes and, similarly, a concept node can have, as children, only dimension nodes.

In addition to nodes, you can use other elements: the parameters, which may arise both from a dimension node (graphically represented by a white square) and from a concept node (white triangle), submitting them to particular constraints. In fact, a concept node can have more than one parameter, while a dimension node can have only a parameter and only in case it has not already children nodes. The introduction of parameters is due to their usefulness in shaping the characteristics that can have an infinite or very high number of attributes. For example, a node representing Cost dimension risks having a high number of values that should be specified by as many concept children nodes. In a similar case, it is therefore preferred to use only one parameter, whose value will be specified in each case. Leaf nodes, in addition to concept nodes, can also be parameters.

In general, each node has a parameter corresponding to a domain, dom(nP). For parameter nodes connected to concept nodes, the domain can be a set of key values from a relational database, while in case of parameter nodes connected to dimension nodes, the domain is a set of possible concept nodes of dimension.

In figure 1, it is shown a general designed CDT, called Meta CDT, which is the starting point for the design of a specific CDT that can be exploited in contextualapplications.

You may note six top dimensions, which correspond to the questions of the 5W1H method: Location (WHERE), Role (WHO), Time (WHEN), Situation (HOW), Interests (WHAT) and Utilization

(WHY).

In particular, there are two types of users and eleven categories of interests.

A context element is defined as an assignment *d_name_i = value*, where d_name_i indicates a possible size or undersize of CDT (it is the label of a dimension node), while *value* may represent the label of one of the concept nodes that are children of the considered dimension node or the value of a parameter referring to one of these concept nodes or the value of a parameter referring to the considered dimension node.

For example, these assignments are possible context elements:

Interest = tourism, Location = LocationID (ID = 3), Role = user, Utilization = holiday.

A context is specified as: $\wedge$ (d_name_i = value).

It is defined as an **"and"** among different context elements.

Several context elements, combined with each other by means of an **"and"**, damage, therefore, the origin of a context.

For example, a possible framework that can be obtained from the previously seen CDT, through the context element that we have listed, is:

C = (Location = locationID (ID=3)) $\wedge$ (Role= user (ID=15)) $\wedge$ (Time = now) $\wedge$ (Situation = routine) $\wedge$ (Interest =tourism) $\wedge$ (Utilization = holiday)

The context is defined as a user, interested in tourism, who uses the contextual app on vacation, in a called place.

Therefore, through the Context Dimension Tree, it is possible, after analyzing the domain of application, to express the size characteristics and values they can take in a graphical way by, respectively, dimension nodes and concept nodes or parameters.

The assignment to a dimension of one of its possible values is a context element. The context element can be considered the main feature of the application, by which a context can be decomposed. The moment you make the formulation of the context, you must specify all the context elements that are part of it and that enable its creation.

Any context is expressible by an **"and"** combination of the context elements to which they are peculiar.

By definition, you can begin to understand how you will create views based on data relating to each context; in fact, they will be built starting from the portions of the database and then from the partial views, associated to the context element that takes part into context information.

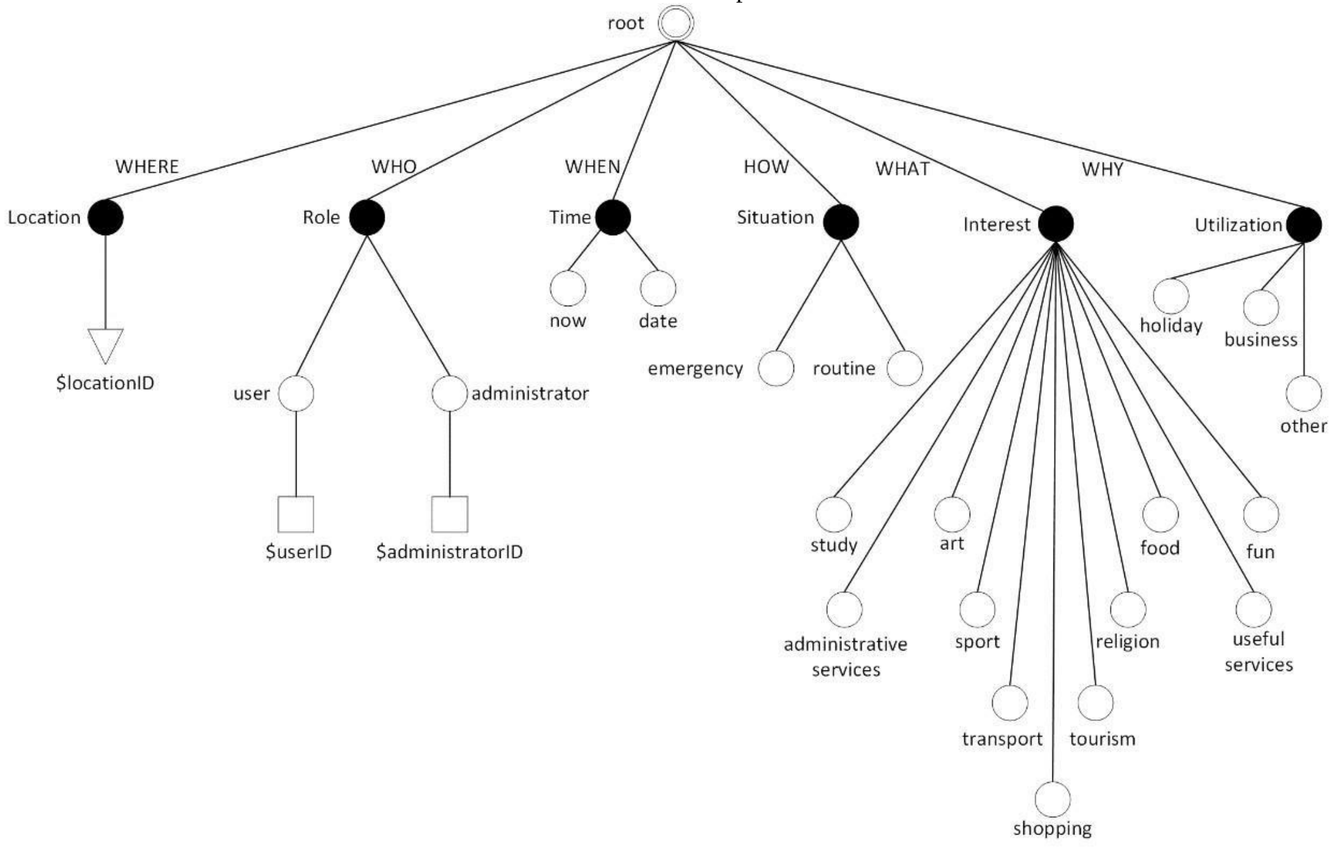


Figure 1. Meta CDT for contextual applications

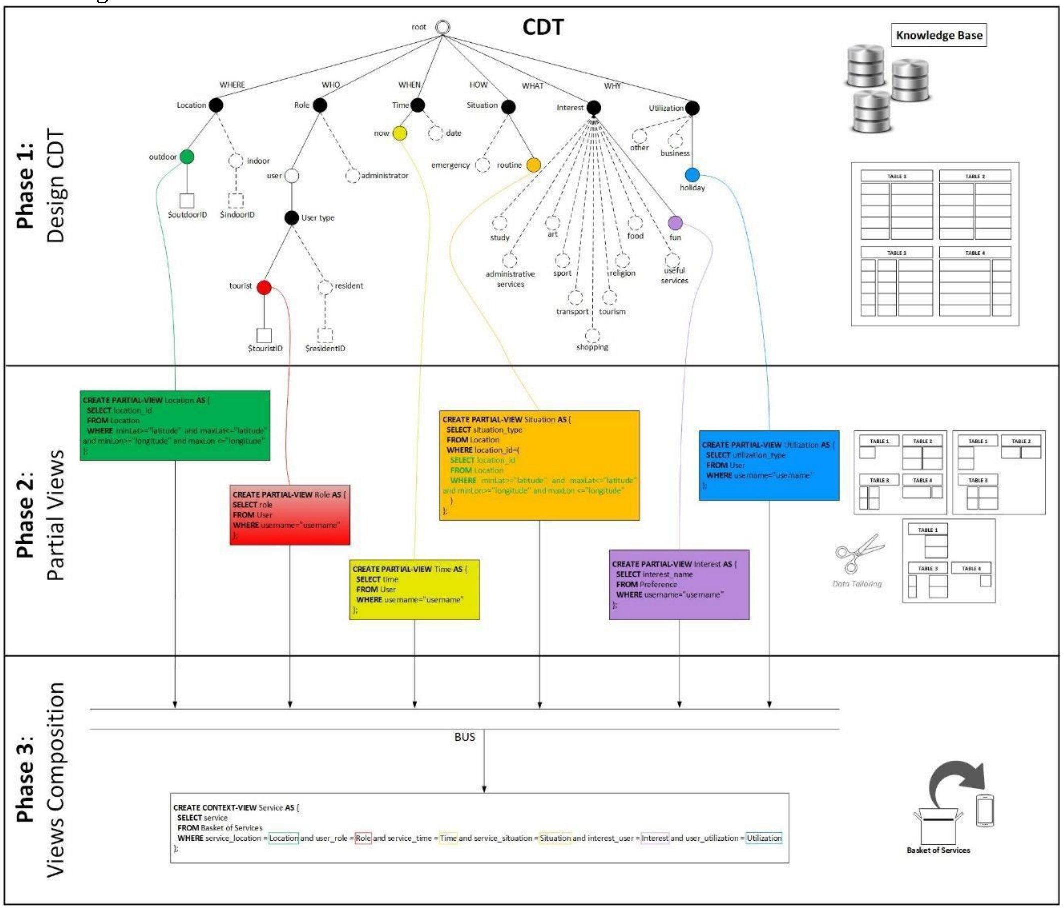
A. *Methodologies and phase to obtain contextual service*

The methodology, shown in figure 2, has been realized in order to manage the database and to carry out reductions of their content based on the context.

The purpose is to help the designer in the definition of all contexts relevant to the considered application and, later, in the association to each context of the portion of the database containing the relevant data about the context.

The methodology consists of three main phases, which we will see in detail later: design phase of the Context Dimension Tree (CDT), definition phase of partial views and composition phase of global views.

1. Design phase of the Context Tree: in this phase, the Context Dimension Tree is designed to identify significant context elements for the considered application. In fact, it focuses on the definition of contexts and on the elements that compose them. These contexts must be identified and shaped, indicating particular elements that characterize each of them. As it has been said, it is available a special tool called Context Dimension Tree (CDT) to make context design. Three CDT were made for specific environments in order to represent and manage a multitude of different contexts and in order to identify, represent, preserve and make available cultural points for each type of user.
2. Definition phase of partial views: after the definition of all the contexts and their context elements, in this step a different portion of the database is associated to each context element, containing the relevant data for it.
   In practice, the goal is to find the appropriate value for a given dimension, in order to obtain, by means of the values of all the dimensions, a valid query and specific to the context in which the user is located.
   A partial view could be related to dimension “Role”: once logged in, the application is able to recognize the user and to know more precisely whether he/she is, for example in tourist areas, a resident or a tourist. Thus, the value “tourist” of dimension “Role” is a partial view for the current context: using this knowledge, you can exclude certain services, not suitable or useful to the tourist role.



```
CREATE PARTIAL-VIEW Location AS {
 SELECT location_id
 FROM Location
 WHERE minLat>="latitude" and maxLat<="latitude"
and minLon>="longitude" and maxLon <="longitude"
};
```

```
CREATE PARTIAL-VIEW Role AS {
SELECT role
FROM User
WHERE username="username"
};
```

```
CREATE PARTIAL-VIEW Time AS {
 SELECT time
 FROM User
 WHERE username="username"
};
```

```
CREATE PARTIAL-VIEW Situation AS {
 SELECT situation_type
 FROM Location
 WHERE location_id={
  SELECT location_id
  FROM Location
  WHERE minLat>="latitude" and maxLat<="latitude"
and minLon>="longitude" and maxLon <="longitude"
  }
};
```

```
CREATE PARTIAL-VIEW Interest AS {
 SELECT interest_name
 FROM Preference
 WHERE username="username"
};
```

```
CREATE PARTIAL-VIEW Utilization AS {
 SELECT utilization_type
 FROM User
 WHERE username="username"
};
```

```
CREATE CONTEXT-VIEW Service AS {
 SELECT service
 FROM Basket of Services
 WHERE service_location = Location and user_role = Role and service_time = Time and service_situation = Situation and interest_user = Interest and user_utilization = Utilization
};
```

Figure 2. General System Workflow

3. Composition phase of global views: this is the phase where you have the automatic generation of views associated with each context, which is made starting from partial views associated with context elements. After the creation of the global views of the contexts, the answers to questions that will be asked to the system will be developed from these views and, in particular, from the view associated with the context in which you are located when the query is performed.

In particular, once defined the values for each dimension, you can use all the information obtained in order to identify the right context and offer data and services customized for the user. It is assumed the example of a tourist who is walking near a beach who gets initially a notification of his/her proximity. Later, he/she needs to deepen such notification. Therefore, it will propose him/her services that they might be interested in, such as the site of the nearest beach, where he/she can get the price list.

B. *System Architecture*

We have made a Context Aware System, whose architecture is shown in figure 3, able to adapt useful data and services to users based on the context. Context awareness of interaction is particularly important in ubiquitous systems and mobile applications for groups of users. In fact, given the ever-increasing variety of interaction devices (fixed and mobile) and application use contexts, it becomes increasingly necessary to develop Context Aware systems that manage information that makes unique and distinguish each human-machine interaction.

The architecture of our model is composed of: the Context Aware Module (CAM), which is the main engine and considers the context in reference to the obtained data (contextual information), in particular position (GPS location), interests and role (obtained during registration) of each user; the Knowledge Base Module, a special type of database for the management of knowledge and information: in particular "Users", representing all users of the application, "Services", which describes all the services of every possible application context, "Resources", which forms all the points of interest and "Events", which describes all events; and finally the Management Module (MM), used both by the administrators of the app and the users themselves.

This module deals with some important issues, including: POIs management, where the insertion can be done directly from map, manually or by search of interests, interacting in the last two cases with Google Maps; services, comments and events management, interacting with TripAdvisor and Facebook/Twitter API.

In figure 4, for a greater immediacy, it is shown a deepening of the architecture realized: the set of user profile, such as preferences and interests, of user context, such as his/her GPS location, of CDT, which provides the rules and allows the representation of the specific context in which he/she is located, of data, including the points of interest and services, allows obtaining the contextual resources tailored for the user, through the use of a contextual application.

On this subject, for the different environments described, we have realized hybrid mobile applications, both in Android and iOS, with many features, some of which are shown in figures 6 and 7: contents, including descriptions, images and services, tailored to interests, profile and location users, planning a route based on user's interests and his/her preferences of travel, exploration of the surroundings from the current position, custom QR Code reader, weather and news on the site, search and insertion of events, the comments section, display position and points of interest on the map, with integration of the navigator on the smartphone to reach specific ones.

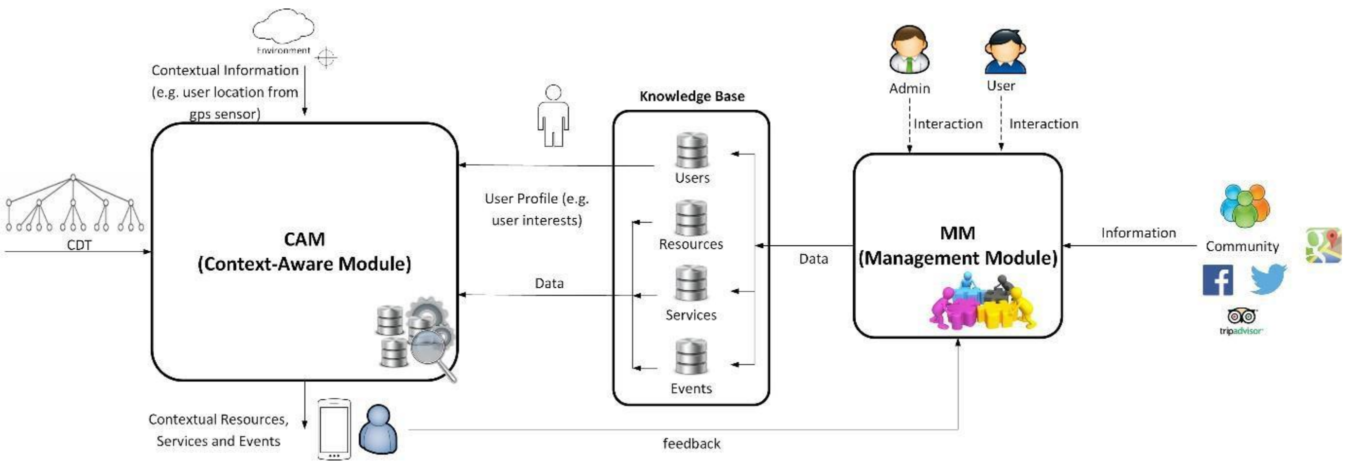

Environment
Contextual Information (e.g. user location from gps sensor)
CDT
CAM (Context-Aware Module)
User Profile (e.g. user interests)
Data
Knowledge Base
Users
Resources
Services
Events
Data
Admin
User
Interaction
Interaction
MM (Management Module)
Information
Community
Contextual Resources, Services and Events
feedback

Figure 3. System Architecture.

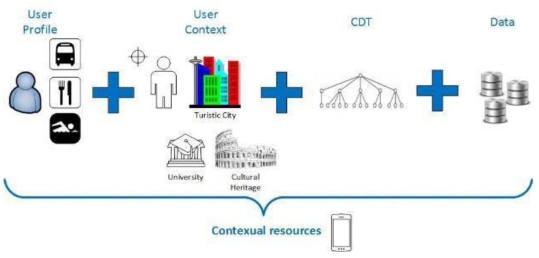


Figure 4. Contextual resources as final result of App.

V. SMARTAPP SALERNO: A CONTEXT AWARE FOR E-TOURISM

In this section, we will present SmartApp Salerno, a contextual app designed and implemented according to what was described previously. In particular, we have thought to apply the approach to the context of the town of Salerno, a municipality in Campania (Italy) of about 135.261 inhabitants and with an extension of 59,85 square km. Along with the Municipality of Salerno, a reference CDT has been designed. In this phase, we have collected the potentially useful services and contents for the citizens and situated them on the map defining the activation zones (fig. 5).

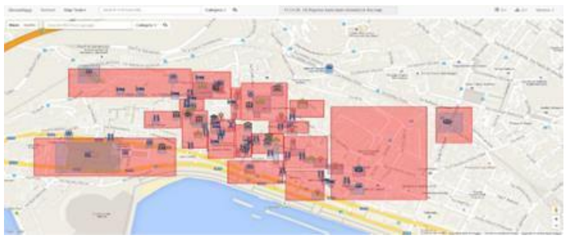


Figure 5. Definition of the activation areas of services and contents.

Moreover, we have defined the different typologies of citizens (elementary school's students, users with kids at school, university students, …) associating them to a previously established set of services and contents. Having the town a series of artistic contents, we have developed services and contents in support of them too. A series of services and contents considered transversal, such as the opening hours of the City Hall, the Library, the Cemetery, the pharmacies, have been made available to all the typologies of users.

All information about places of worship and shops has been uploaded, for any building or area of potential.

The App has been developed with hybrid technologies (Cordova and PhoneGap) to allow an easier publication both in Android and Apple environment.

The App has been presented to various tourists in December 2015 and January 2016 during 30 workshops. The occasion has been given by a Christmas event, called Luci di Artista (Artist's Lights), that every year is held in Salerno and that involves hundreds of thousands of tourists. They have been involved overall about 3000 tourists between 18 and 50 years old. During each workshop, the app has been installed on the mobile devices of the tourists. For each of them the system has started to supply personalized itineraries (figure 6).

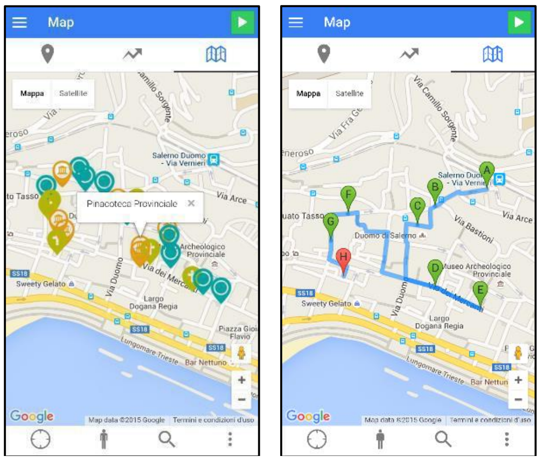


Figure 6. Screenshots with some features of contextual application.

After a week the system has given a questionnaire of five sections to each tourist. To every question present in the section, five possible answers have been associated: I totally agree – I agree – Undecided – I disagree- I totally disagree. The questionnaire in detail is the following:

Section A: SmartApp Salerno – Context

- A1. SmartApp gives the user tailor- made contents and services
- A2. SmartApp allows the user to know several points of interest of the Old Town Center of Salerno
- A3. SmartApp supplies contents and services in the right place
- A4. SmartApp supplies services according to the interests selected in the user profile

Section B: SmartApp Salerno – Usability

- B1. SmartApp is immediate to understand and use
- B2. The registration is quick to do and non- invasive

Section C: SmartApp Salerno – Further aspects

- C1. Information about each point of interest is very useful
- C2. I do not know other applications like SmartApp
- C3. The contents, such as descriptions and images, are of high quality and represent one of the strong points of SmartApp
- C4. The services associated to the points of interest allow a higher immediacy than a classic research on the Internet

Section D: SmartApp Salerno – Functionality

- D1. The map is very useful and well curated
- D2. The plan itinerary service allows easily realizing an itinerary in the Old Town Center of Salerno according to the user's preferences
- D3. The explore surroundings service is very useful to know what there is nearby and eventually reach them
- D4. The functionality of research of points of interest by category of interest is intuitive and practical
- D5. It is useful to know if a certain point of interest is open or closed
- D6. The functionality of QR code in inner environments can be well used
- D7. The tutorial effectively allows learning the main characteristics of SmartApp

- D8. The weather forecast and the news are two very useful services

Section E: SmartApp Salerno – Future developments

- E1. It would be interesting to have a higher integration with the main social networks
- E2. It would be interesting to insert the available time in the plan itinerary service

As can noticed from the figure 7, users show great appreciation for the app. In general, they appreciated the proposed contents and services.

VI. Conclusions

In this paper, we have presented an app able to offer services and contents personalized for the needs of the user according to the context where he/she is. The app bases its 'contextual' functioning on the adoption of the CDT that is able to shape the context and the actions to implement. The app has been developed for the needs of a little Italian town and the first results have been satisfying. The following activities have as purpose the application of the proposed methodology to more complex environments, for dimension and number of potential points of interest to manage.

Acknowledgements

The research reported in this paper has been supported by the Project Cultural Heritage Information System (CHIS) PON03PE_00099_1 CUP E66J140000 70007 – D46J1400000 0007 and the Databenc District.

References

[1] L. Atzori, A. Iera, Morabito, "The internet of things: A survey," *Computer Networks*, 54(15), 2010, pp. 2787–2805.

[2] J. M. Hernandez-Munoz, et al., "Smart cities at the forefront of the future internet," *Future Internet Assembly*, LNCS, 6656, 2011, pp. 447–462,.

[3] H. Schaffers et al., "Smart cities and the future internet: Towards cooperation frameworks for open innovation," *Future Internet Assembly*, LNCS, 6656, 2011, pp. 431–446.

[4] N. Komninos, H. Schaffers, M. Pallot, "Developing a policy roadmap for smart cities and the future internet." In *eChallenges e-2011 Conference Proceedings*, 2011, pp. 286-306.

[5] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, L. Tanca, "Context information for knowledge reshaping," *Int. J. Web Eng. Technol.* 5(1), 2009, pp. 88-103.

[6] C. Bolchini, C. Curino, F. A. Schreiber, L. Tanca, "Context integration for mobile data tailoring," *SEBD 2006*, 2006, pp. 48-55.

[7] F. Colace, L. Greco, S. Lemma, M. Lombardi, Duncan Yung, Shi-Kuo Chang, "An Adaptive Contextual Recommender System: a Slow Intelligence Perspective", SEKE 2015: 64-71

[8] Francesco Colace, Massimo De Santo, Luca Greco, Vincenzo Moscato, Antonio Picariello: A collaborative user-centered framework for recommending items in Online Social Networks. Computers in Human Behavior 51: 694-704 (2015)

[9] Francesco Colace, Massimo De Santo, Luca Greco: An adaptive product configurator based on slow intelligence approach. IJMSO 9(2): 128-137 (2014)

[10] F Colace, V Moscato, E Quintarelli, E Rabosio, L Tanca, Context awareness in pervasive information management, Data Management in Pervasive Systems, 235-256 3, 2015

[11] F Colace, M De Santo, V Moscato, A Picariello, FA Schreiber, L Tanca, PATCH: A Portable Context- Aware ATlas for Browsing Cultural Heritage Data Management in Pervasive Systems, 345-361, 2015.

[12] Bitner, M.J., Faranda, W.T., Hubbert, A.R., Zeithaml, V.A., 1997. *Customer Contributions and Roles in Service Delivery*. International Journal of Service Industry Management, Vol.8, No.3, pp. 193-205.

[13] Ciasullo M.V., Troisi O., 2013. *Sustainable value cration in SMEs: A case study*. The TQM Journal, Vol.25, No.1, pp. 44-61.

[14] Colace, F., Greco, L., Lemma, S., Lombardi, M., Amato, F., Moscato, V., Picariello, A., 2015f. *Contextual Aware Computing and Tourism: A Case Study*. The Eleventh International Conference on Signal- Image Technology & Internet-Based Systems (SITIS), pp. 804-808.

[15] Dey, A.K., Abowd, G.D., 1999. *Towards a Better Understanding of Context and Context-Awareness*. HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, pp. 304-307.

[16] Ryan, N., Pascoe, J., Morse, D., 1997. *Enhanced Reality Fieldwork: the Context Aware Archaeological Assistant*. Computer Applications and Quantitative Methods in Archaeology. Proceedings of the 25th Anniversary Conference, Archaeopress, Oxford, pp. 269-274.

[17] Schilit, B., Adams, N., Want, R., 1994. *Context- Aware Computing Applications*. Proceedings Of The Workshop On Mobile Computing Systems And Applications, pp. 85-90.

[18] Tanca, L., Bolchini, C., Curino, C., Schreiber, F.A., 2006. *Context integration for mobile data tailoring*. Italian Symposium on Database Systems (SEBD), pp. 48-55.

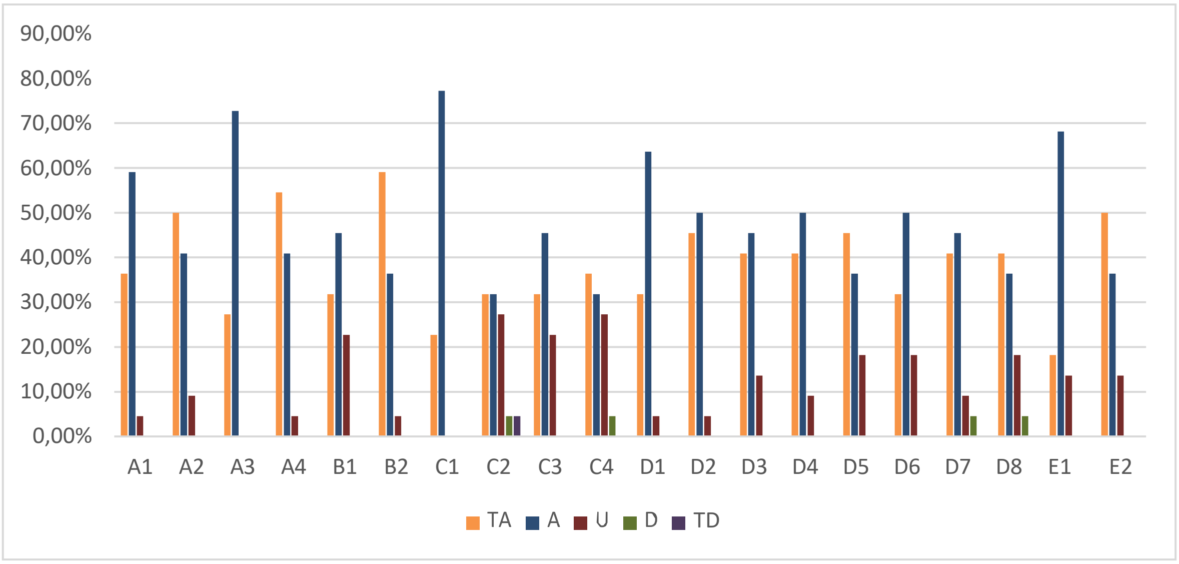


Figure 7. Analysis of Questionnaires (TA I totally Agree – A Agree – U Undecided – D I disagree – TD I totally disagree).

A Mobile TDR System for Smart Phones

Shi-Kuo Chang, Wei Guo, Duncan Yung, ZiNan Zhang, HaoRan Zhang and WenBin You
Department of Computer Science
University of Pittsburgh, Pittsburgh, PA 15238, USA
{schang, weg21, kay35, haz64, ziz22}@pitt.edu and youwenbin@nuc.edu.cn

***Abstract*—In our previous work a multi-level slow intelligence system with multiple sensors, called the TDR system, was developed [1]. It consists of interacting super-components each with different computation cycles specified by an abstract machine model. The TDR system has three major super-components: Tian (Heaven), Di (Earth) and Ren (Human), which are the essential ingredients of a human-centric psycho-physical system following the Chinese philosophy. Each super-component further consists of interacting components supported by an SIS server. In this paper we further developed a mobile TDR system for smart phones, intended for practicing health exercises such as conducting meditation. The initial experimental results and further research topics are discussed.**

Keywords—slow intelligence system, component-based software engineering, sensor networks, mobile TDR system.

1. Introduction

Our goal is to develop a mobile TDR system for smart phones so that the user can carry this mobile TDR system anywhere. This experimental TDR system thus provides a platform for exploring and integrating different applications in personal health care, emergency management and social networks.

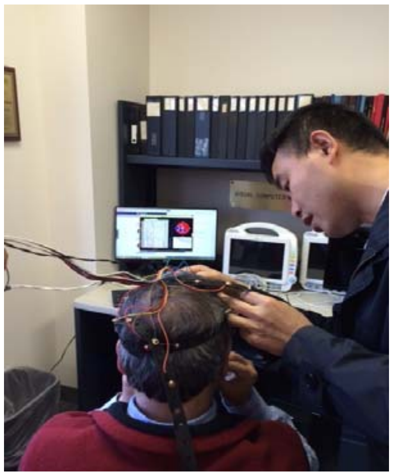


Figure 1. A brain wave headset.

DOI reference number: 10.18293/DMSVLSS2017-016

To develop this experimental mobile TDR system, we need to port the TDR system to a smart phone. This mobile TDR system empowers the user so that he or she can have continuous access to the sensory devices and apps offered by the mobile TDR system. An example of such a sensory device is the brain wave headset. As shown in Figure 1, the user can wear the headset with sixteen OpenBCI brain wave sensors so that the time signals detected by the sensors can be continuously sent to the TDR system. The displayed time signals (left), positions of the sensors (upper right) and power spectrum (lower right) are illustrated in Figure 2.

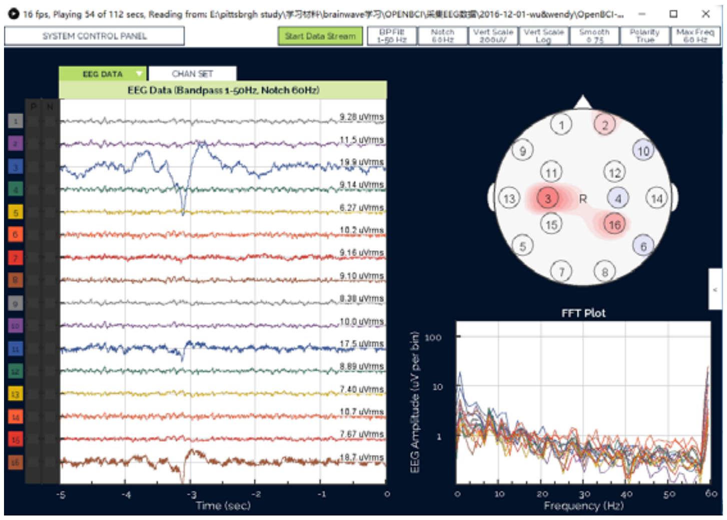


Figure 2. Time signals from OpenBCI brain wave sensors.

In addition to brain wave sensors, a smart phone has built-in sensors such as its camera, which can be used to analyze the gaze of the user (see Figure 3) to obtain certain measurements. The smart phone also has audio output. Thus the mobile TDK system provides multiple sensory input/output devices to continuously monitor user's state of health. In our initial experiment, the objective is to monitor user's meditation state.

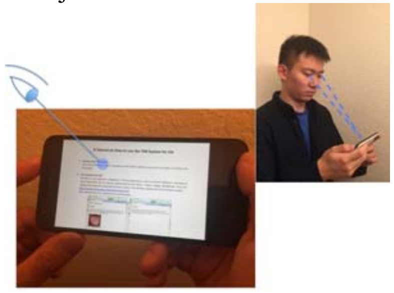


Figure 3. Gaze analysis.

We need to answer two basic questions: (1) Can a mobile TDR system for smart phone be developed? (2) Can the information from various sensors of the mobile TDR system be analyzed and combined to monitor user's meditation state?

The paper is organized as follows. To answer the first question, Section 2 presents the system architecture and development environment. The basic scenarios of the mobile TDR system are described in Section 3.

To answer the second question, the identification of meditation state from brain wave sensors is described in Section 4. Gaze analysis for the detection of meditation state is described in Section 5. Experimental results show the two approaches are consistent in detecting the meditation state.

The related work is reviewed in Section 6. Section 7 discusses further research topics.

2. System Architecture

To facilitate the design of complex slow intelligence systems such as human-centric psycho-physical systems, we introduced the concept of super-components [1]. A complex slow intelligence system basically consists of interacting super-components, which further consist of many interacting components supported by an SIS server. Communications are through the SIS server, and the messages are *layered*, i.e., each message type has its hierarchical *scope*. A super-component can be viewed as a collection of components interacting by messages within the same scope. From an architectural viewpoint the result is a multi-level slow intelligence system as illustrated by Figure 4.

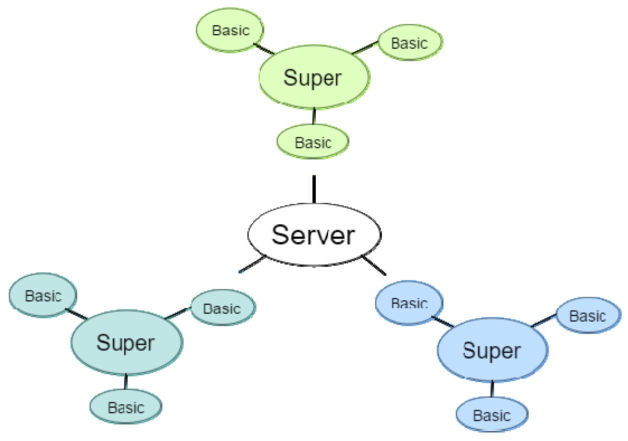


Figure 4. A multi-level slow intelligence system.

The TDR System is such a multi-level slow intelligence system. Figure 5 shows the structure of the TDR System. The seven components are the graphical user interface **PrjRemote**, **Audio** output, **Gaze** analyzer, internet **Uploader**, **OpenBCI** brain wave input processor, and data **Filter**. These components communicate with each other via the **Server**. These seven components all reside in the smart phone so that the mobile TDR system can quickly respond to sensory input. For this reason a high-end smart phone, ASUS ZenFone 3 ZS570KL with 64GB storage, 6 GB RAM, micro-SDXC Memory Card Slot, USB Type-C and Android 6.0 was acquired.

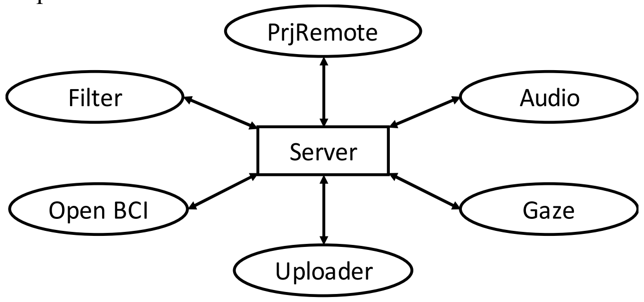


Figure 5. The structure of the mobile TDR system.

The development environment is illustrated by the UML development diagram shown in Figure 6. The Android components are first developed in Android Studio on a PC and then uploaded to the smart phone. The OpenBCI brain wave headset is the external device connected to the smart phone via Blue Tooth and a USB port. As shown in Figure 6 the headset can also be augmented for audio output such as rhythmic music (see Section 7).

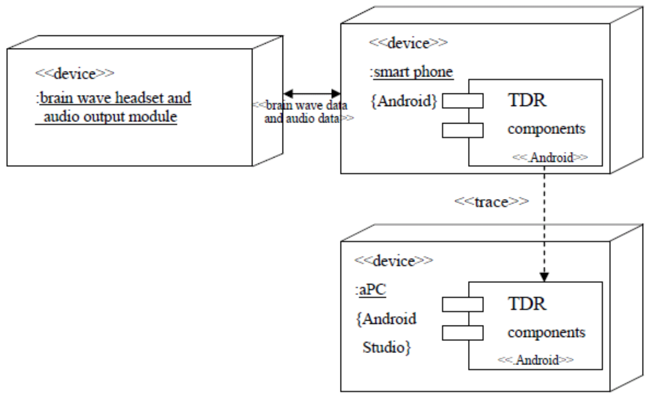


Figure 6. TDR Development environment.

\

3. Basic Scenarios of Mobile TDR System

The mobile TDR system initially provides the following apps: (1) The Audio app plays rhythmic music to help calm down the user. (2) The Gaze app analyzes a user's gaze to decide whether he/she is in meditation state. (3) The BrainWave app analyzes the brain wave signals from the brain wave headset if the user is wearing, and uploads data to Internet for further processing. (4) The Comparator compares the decisions made by the Gaze analyzer and the Brain wave app in predicting meditation state.

The first three apps are in the smart phone, and the Comparator may either be in the smart phone or runs on the

Internet. The latter option allows the continuous improvement of the analyzer so that it can learn and improve its performance following slow intelligence principles.

3.1. The Audio App

This app requires three components: PrjRemote, Server and Audio. The Audio component can play a small piece of rhythmic music (e.g. raining) again and again at a certain frequency, helping user to relax. There are several pieces of music pre-stored in the Audio component. User can select or change the music by sending an ID number from PrjRemote. PrjRemote will send the ID number to the Server, Server will forward the ID number to Audio and finally Audio will play the music that user selected, as shown in Figure 7.

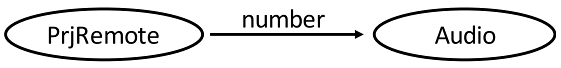


Figure 7. Information flow for Audio App.

For example, user wants to listen to the second piece of music. User types “2” in the PrjRemote and clicks the button “send” (Figure 8 (a)). Then, user hears the Audio component playing the number “2” music and Audio component displays the received message on its interface (Figure 8 (b)).

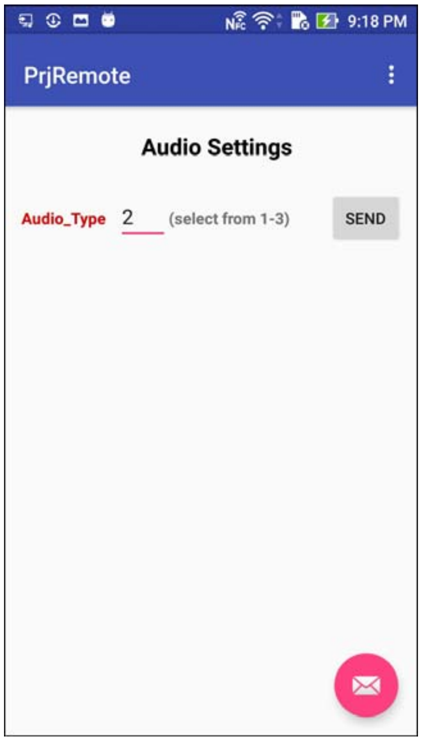
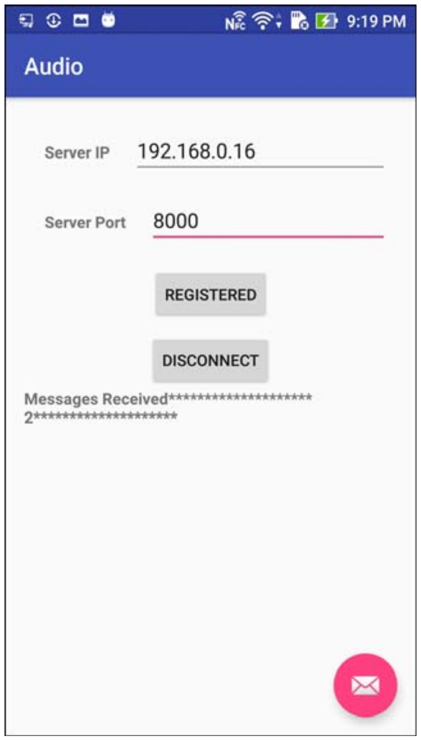


Figure 8(a). Audio_type “2” is selected.

3.2. The Gaze App

This app requires three components: PrjRemote, Server and Gaze Analyzer, as shown in Figure 9. The Gaze Analyzer component evaluates whether the user is relaxed or not by analyzing user’s gaze. After the analysis, Gaze will generate a parameter to reflect the degree of the user’s relaxation. When the Gaze component is activated, it will open the front camera of the smart phone and record a video clip of the user’s face. The duration of the recording is set by the user by sending a number from PrjRemote to Gaze.



Messages Received*******************
2*******************

Figure8(b). Audio component received “2”.

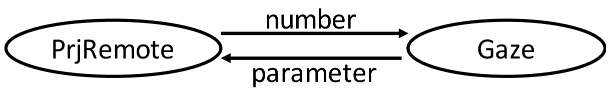
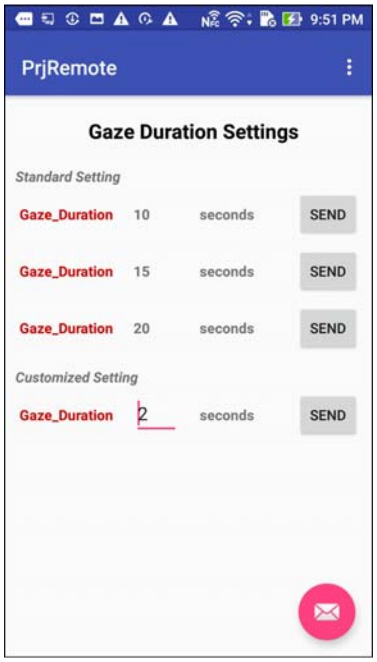


Figure 9. Information flow for the Gaze sub-system.

In Gaze Duration Setting view of PrjRemote user can choose standard setting or customized setting. For example, user wants to set the Gaze duration to be 10 seconds. He clicks the button “send” on the first line of Standard Setting (Figure 10 (a)).

When the Gaze component receives the message, it changes the duration time and displays the message on the bottom (Figure 10 (b)).

Next, user activates the Gaze component by clicking the ”Start Mental State Tracking” button. After the recording and the gaze analysis. the result parameter “0.75” is displayed on the bottom of the PrjRemote interface (Figure 10 (c)).


9:51 PM

PrjRemote

Gaze Duration Settings

Standard Setting

Gaze_Duration 10 seconds SEND

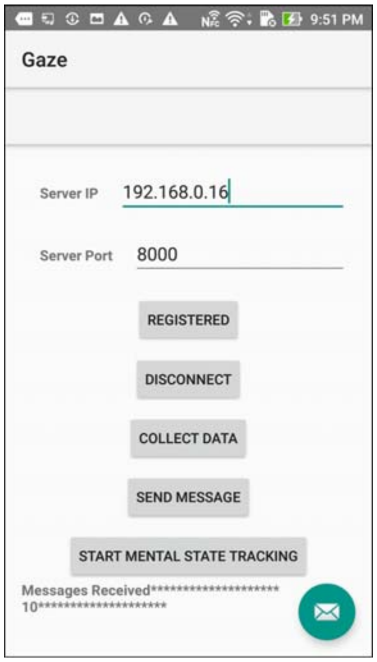
Gaze_Duration 15 seconds SEND

Gaze_Duration 20 seconds SEND

Customized Setting

Gaze_Duration 2 seconds SEND

Figure 10(a). Gaze_duration “10” is selected.


9:51 PM

Gaze

Server IP 192.168.0.16

Server Port 8000

REGISTERED

DISCONNECT

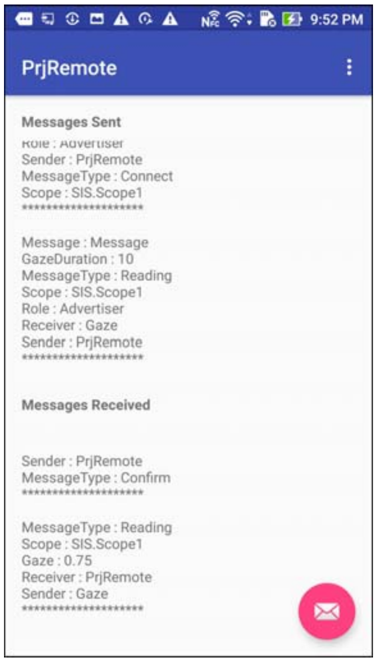
COLLECT DATA

SEND MESSAGE

START MENTAL STATE TRACKING

Messages Received********************
10********************

Figure 10(b). Received message is displayed.


9:52 PM

PrjRemote

Messages Sent

Role : Advertiser
Sender : PrjRemote
MessageType : Connect
Scope : SIS.Scope1
****

Message : Message
GazeDuration : 10
MessageType : Reading
Scope : SIS.Scope1
Role : Advertiser
Receiver : Gaze
Sender : PrjRemote
****

Messages Received

Sender : PrjRemote
MessageType : Confirm
****

MessageType : Reading
Scope : SIS.Scope1
Gaze : 0.75
Receiver : PrjRemote
Sender : Gaze
****

Figure 10(c). Gaze parameter “0.75” is displayed.

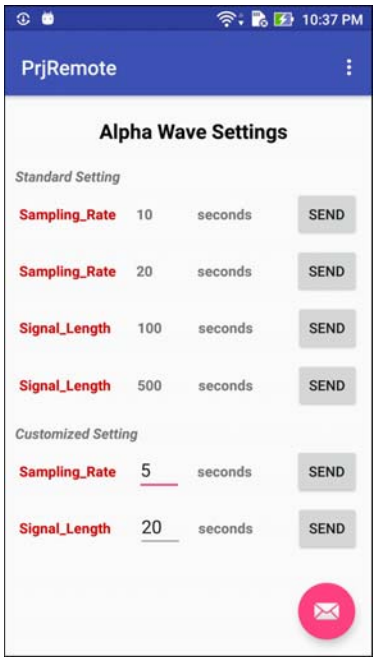
3.3. The Brain Wave App

This app requires five components: Filter, Open-BCI, Uploader, PrjRemote and Server. The Open-BCI input processor component receives the input data from a wearable brain wave headset. User can set the frequency and duration of the input data collected from the wearable headset by setting these two parameters.

$$\begin{tikzcd}
\text{Open BCI} \arrow[r, "\text{data}"] & \text{Filter} \arrow[r, "\text{data}"] & \text{Uploader} \\
 & \text{PrjRemote} \arrow[u, "\text{parameter}"] &
\end{tikzcd}$$

Figure 11. Information flow for Brain Wave App.

Suppose the user wants to collect the data every 5 seconds, and the total time period for collecting data is 20 seconds. He enters “5” on the first line of Customized Setting and enters “20” on the second line, and then clicks the “send” button (Figure 12(a)).



10:37 PM

PrjRemote

Alpha Wave Settings

Standard Setting

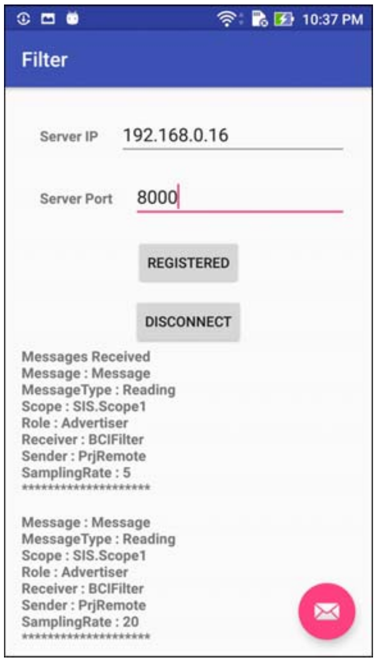
Sampling_Rate	10	seconds	SEND
Sampling_Rate	20	seconds	SEND
Signal_Length	100	seconds	SEND
Signal_Length	500	seconds	SEND

Customized Setting

Sampling_Rate	5	seconds	SEND
Signal_Length	20	seconds	SEND

Figure 12(a). Frequency and time period are selected.

When the filter component receives the frequency and time period from PrjRemote, it displays them on its interface (Figure 12 (b)). Open-BCI input processor receives data that are filtered according to the parameters set by the user (Figure 12(c)).



10:37 PM

Filter

Server IP 192.168.0.16

Server Port 8000

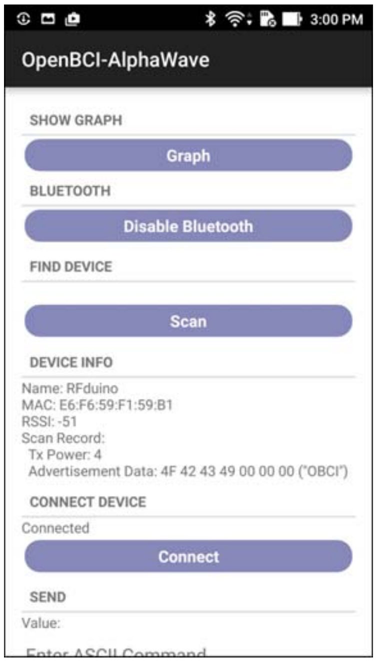
REGISTERED

DISCONNECT

```
Messages Received
Message : Message
MessageType : Reading
Scope : SIS.Scope1
Role : Advertiser
Receiver : BCIFilter
Sender : PrjRemote
SamplingRate : 5
********************

Message : Message
MessageType : Reading
Scope : SIS.Scope1
Role : Advertiser
Receiver : BCIFilter
Sender : PrjRemote
SamplingRate : 20
********************
```

Figure 12(b). Filter is activated.



3:00 PM

OpenBCI-AlphaWave

SHOW GRAPH

Graph

BLUETOOTH

Disable Bluetooth

FIND DEVICE

Scan

DEVICE INFO

```
Name: RFduino
MAC: E6:F6:59:F1:59:B1
RSSI: -51
Scan Record:
 Tx Power: 4
 Advertisement Data: 4F 42 43 49 00 00 00 ("OBCI")
```

CONNECT DEVICE

Connected

Connect

SEND

Value:

Enter ASCII Command

Figure 12(c). Data is received and filtered.

The OpenBCI input processor can process and plot the input Alpha brain wave data as shown in Figure 12(d). When the user is concentrating, the curve tends to move upward and the *alert probability* increases. In Figure 12(d) the alert probability is 0.8 (80%). When the user is not concentrating, the curves tends to move download and the alert probability decreases. Finally the uploader can upload the data to the Internet and store it in a database.

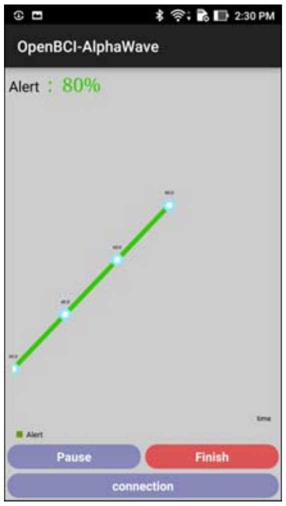


Figure 12(d). Alpha Brain Wave data is plotted.

3.4. The Comparator App

After the gaze data and brainwave data have been uploaded to the database in the Internet, they can be further analyzed to decide whether the analyses made by the Gaze Analyzer and the Brain Wave App are consistent. The results are illustrated in Figure 16, where the horizontal axis can be the time line. The green color prediction is when both EEG and eye-tracking predictions are consistent and correct. Blue colors are brain wave predictions that are inconsistent with eye-tracking predictions. The red colors are eye-tracking predictions that are incorrectly predicted. 1 is meditation state, -1 is high cognitive workload.

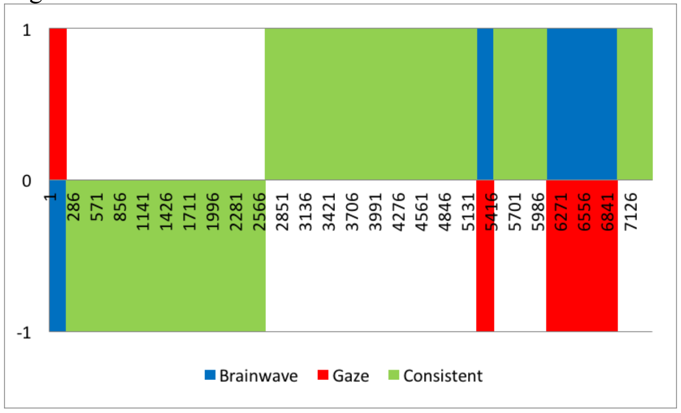


Figure 13. Comparison of meditation state predictions.

4. Meditation State Detection

Meditation, according to [2], is used to describe "practices that self-regulate the body and mind, thereby affecting mental events by engaging a specific attention set." In the Western tradition, meditation can be classified into *mindfulness meditation* and *concentrative meditation* [2]. Mindfulness meditation emphasizes the rise of all possible feelings in awareness, whereas concentrative meditation requires concentrating on a specific object or activity.

In our initial experiment we regard meditation as concentrative meditation. Our concentrative meditation system includes three major components: 1) State-Meditating: detecting concentrative meditation state via brainwave signals; 2) State-Meditating: detecting concentrative meditation state via eye tracking signals; 3) Trait-reading: detecting concentrative meditation traits via eye movement patterns.

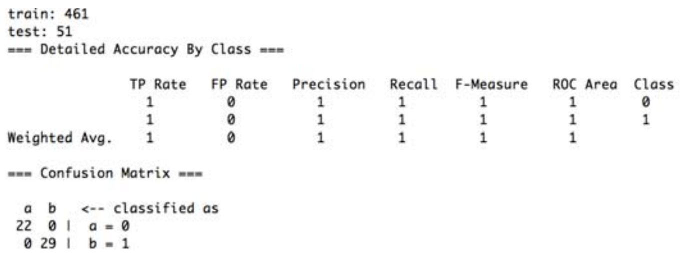
State-Meditating: Brainwave

According to [2], the meditation state is the state when the user is more relaxed as exhibited by the brain wave patterns with little or no activity. Given the strong relationship reported between EEG signal and meditation state, we developed a technique to detect the meditation state, or more precisely, to predict the probability of the meditation state, from input brain wave data.

The alert probability described in the previous section can be regarded as a measurement of the meditation state probability from AlphaWave provided by OpenBCI software.
In our approach, The OpenBCI Monitor component makes such prediction based on pre-trained prediction model, and decides whether it is necessary to upload data into the database for further analysis.
For training the prediction model, a software tool called "Weka" is used. Weka provides a lot of flexible well-programmed machine learning algorithms. With its help, the module can train a prediction model easily using different machine learning algorithms. In addition, Weka is a package implemented in Java, so it is easy to integrate the package into the TDR system as a component.

After the 16 channel brain wave data is appropriately cleaned, 4 consecutive records of brain wave data, plus the mean, variance and standard deviation of these records, are used to train the predication model. Weka provides a lot of machine learning algorithms, from KNN or linear regression, to other high level machine learning algorithms, such as SVM. With the help of the package, the system can train any model by different classifiers. We implemented the program that, user can easily to change the classifier they need. To select a proper algorithm, we did a few experiments using different classifiers, such as Linear Regression, Logistic Regression, Bagging, Ada Boost, Naïve Bayes, J48, Random Forest, and SMV. We used 10-folds cross validate to select a better model. In our initial experiment, the two subjects are both experts in meditation, therefore no matter what classifier was used, the prediction accuracy is always 100%. Therefore we chose to use SMV as the model.

To summarize our initial experiment, we can train, test a model, and save the model into a file so that the BCI Monitor can load the model easily and make prediction dynamically in a live demo. We did a pilot study to test the workability of our system. In the pilot study, we recruited a concentrative meditating master, to meditate (s1) and stay calm (s2). The experimental results are shown in Figure 14.


```
train: 461
test: 51
=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
                   1         0          1          1          1          1         0
                   1         0          1          1          1          1         1
Weighted Avg.      1         0          1          1          1          1

=== Confusion Matrix ===

  a  b   <-- classified as
 22  0 |  a = 0
  0 29 |  b = 1
```

Figure 14. Experimental results of meditation state prediction.

We can see the predication accuracy is always 100%. Such results are of course too good to be true. We are curious whether the mastery of meditation induces the huge difference in the two states. Therefore we design another pilot study to evaluate it, which will be described in later section.

State-Meditating: Eye-Tracking

When a user is walking or exercising, it is inconvenient to wear a brain wave headset with many electrodes. Therefore,

we would like to explore whether the meditation state can be detected through gaze analysis. If it can be done, then the user only needs his smart phone and nothing else.
We propose to use face-tracking and eye-tracking technique to monitor meditating on smart phone based on two hypothesis: 1) when a user is meditating, there is a potential that the mental and conscious change might affect the user's facial emotions slightly; 2) When a user is meditating, the conscious change might affect his eye-movements even if the eyes are closed. We are investigating the potential to observe meditating state via appearance changes. Our State-Meditating function is designed to be an application connected with SIS-server. We used a Google Nexus 5x smart phone running Android 6.0 to launch the SIS system applications. After the user settled the duration parameter on SIS-GUI, SIS-GUI will redirect the user to Gaze-component. The meditating state tracking will be started after user clicked the start button. During state tracking, each frame captured by the front facing camera will go through 2 steps: 1) Face landmarks tracking: we track the important landmarks of a face (e.g. left, right, and center point of eyebrows). In table 1, we listed the detail face landmarks being tracked; 3) Eye gaze estimation: we rely on the location of the pupil relative to the rest of the eye to estimate the direction of eye gaze. In our Gaze-Component, the Qualcomm Snapdragon SDK is being used to accelerate the tracking process. On 808 CPU in Nexus 5x, the per-frame image processing time is 17 ms.

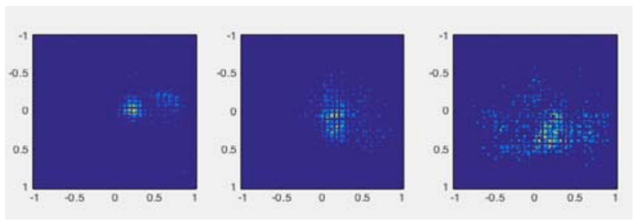
Region	Landmarks Detail
Brows	leftEyeBrowsPointTop
	leftEyeBrowsPointBot
	leftEyeBrowsPointLeft
	leftEyeBrowsPointRight
	rightEyeBrowsPointTop
	rightEyeBrowsPointBot
	rightEyeBrowsPointLeft
	rightEyeBrowsPointRight
Ear	leftEarPointTop
	leftEarPointBottom
	rightEarPointTop
	rightEarPointBottom
Eye	leftEyeBot
	leftEyeTop
	leftEyeCenter
	leftEyeLeft
	leftEyeRight
	rightEyeBot
	rightEyeTop
	rightEyeCenter
	rightEyeLeft
	rightEyeRight
Mouth	mouthULipBot
	mouthULipTop
	mouthLLipBot
	mouthLLipTop
	mouthLeft
	mouthRight
Nose	noseBridgePoint
	noseCenterPoint
	noseLLeft
	noseLRight
	noseMLeft
	noseMRight
	noseTipPoint
	noseULeft
	noseURight

Table 1. Details of face tracking landmarks.

Pilot study

We design another pilot study to 1) evaluate the accuracy of state prediction, 2) test whether our system is applicable to non-professional users (which are our targeted users), 3) compare the signals of brainwave and eye tracking in state prediction.

In this pilot study, we recruited a local college master student (male, age 24) to test the meditation state prediction accuracy. The subject performed 3 different state in front of a mobile phone front facing camera with EEG sensors on: concentrative meditation (s1), stay calm (s2) and high cognitive workload (s3). In s1, we followed concentrative meditation instruction to train the user to focus on his nose and try to arise any feeling on nose. In s2, the subject simply stayed calm and avoided thinking anything. In s3, we ask the subject to thinking a math problem (a four digit number plus a four digit number) to induce the cognitive workload.



(a) (b) (c)

Figure 15. Eye gaze heat map on different state: (a) s1, meditation state; (b) s2, staying calm state; (c) s3, high cognitive workload state. We observe a clear difference among different state, where s1 is the most concentrative state and s3 is the sparsest state.

We use OpenBCI to collect EEG data, and smart phone front facing camera to collect eye-tracking signals. We parallelize the two groups of data, and separate both EEG data and eye-tracking data into 2 parts: first 70% as training and later 30% as testing.

To train the EEG model, we followed the previous pilot study and used Weka in java, and SVM for creating the model. The features we extracted are 16 channel brain wave data and the mean, variance and standard deviation of these records within 4 step sliding window.

From the raw gaze data, we can observe a clear difference among different state, where s1 is the most concentrative state and s3 is the sparsest state (Figure 15). To quantify the difference, we also trained the eye-tracking model. We kept consistent with EEG model, using Weka in java to perform SVM, with gaze x and gaze y coordinate at each timestamp and the corresponding mean, variance and standard deviation within 2s sliding window.

When comparing concentrative meditation (s1) with calm state (s2), the EEG model for predicting meditation state is 100% accuracy and eye-tracking model is 82.86% accuracy. The corresponding parallel result is shown in Figure 16 (which is the same as Figure 13).

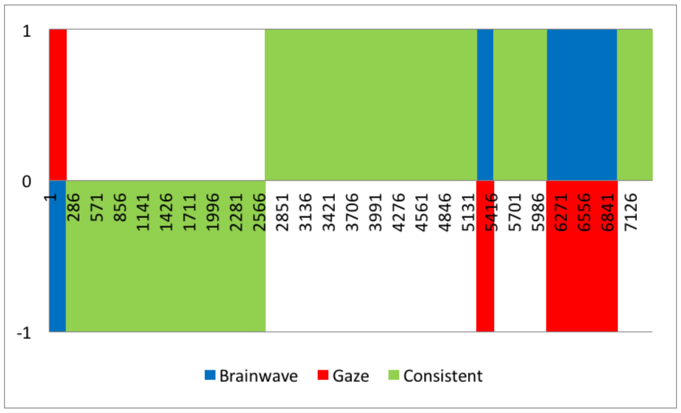


Figure 16. Meditation prediction from staying calm states. The green color prediction is when both EEG and eye-tracking predictions are consistent and correct. Other than green, blue colors are EEG predictions that are inconsistent with eye-tracking predictions. The red colors are eye-tracking predictions that are incorrectly predicted. 1 is meditation state, -1 is staying calm state.

When comparing concentrative meditation (s1) with high cognitive workload state (s3), the EEG model for predicting meditation state is 100% accuracy and eye-tracking model is 97.06% accuracy. The corresponding parallel result is shown in Figure 17.

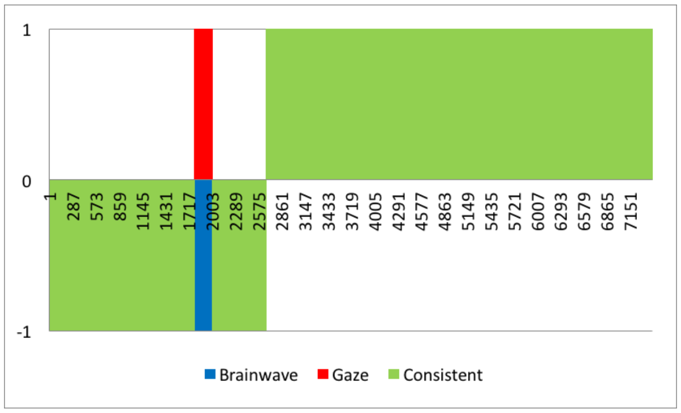


Figure 17. Meditation prediction from high cognitive workload states. The green color prediction is when both EEG and eye-tracking predictions are consistent and correct. Other than green, blue colors are EEG predictions that are inconsistent with eye-tracking predictions. The red colors are eye-tracking predictions that are incorrectly predicted. 1 is meditation state, -1 is high cognitive workload.

5. Detecting Reading Patterns by Gaze Analysis

In this section, we first explain how to apply gaze analysis to find out user's reading patterns. This will give some empirical justification that such technique may be applicable to track user's reading patterns to determine users' meditation trait. For initial design, we use the instruction manual to practice Chi as the reading material, and track users' gaze pattern to see whether the user understands the instruction. We believe the time a user takes to read and the user's gaze pattern reflect his (her) meditating trait.

To process the data in SIS database and visualize the processing workflow, we write our code in PHP, and the visualization can be easily observed on SIS GUI webpage.
The visual object consists of two levels. Using visual object definition, the two levels are:

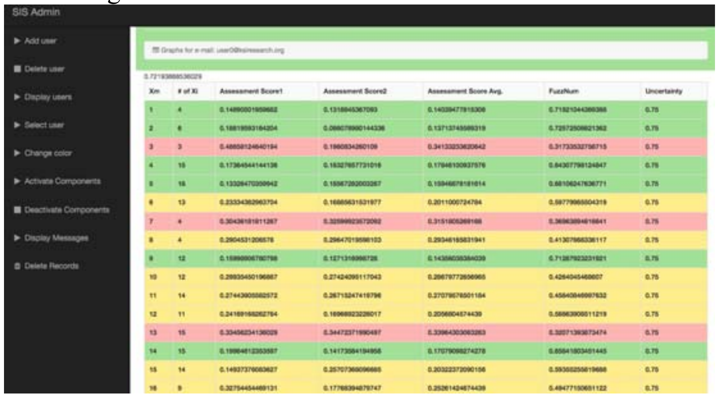
Top level:
Y_m is: Binary Trait State (Have trait of meditation or not)
Y_i is: Different color on 'Reading' Tag (Red: don't have trait, Green: have trait)
Bottom Level:
X_m is: User's Gaze data within 1 second time period
X_i is: Gaze coordinates (points) and corresponding time
Our system controlled by two directions: bottom-up and top-down. The top down direction is on scope of TDR system in the whole.
In bottom up direction:
H_i represents the covariance value for each individual segment X_m calculated by function:

$$\operatorname{cov}(X,Y) = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{1}{2}(x_i - x_j)\cdot(y_i - y_j) = \frac{1}{n^2}\sum_{i}\sum_{j>i}(x_i - x_j)\cdot(y_i - y_j).$$

We set a fuzzy number (fuzz) R ranging from 0 to 1 with 5 levels. If Hi = 0-0.2, then the fuzzy number R=1, the level indicates the user has a strong trait; If Hi = 0.2-0.4, then the fuzzy number R=0.8, the level indicates the user has a trait; If Hi = 0.4-0.6, then the fuzzy number R=0.6, the level is medium; If Hi = 0.6-0.8, then the fuzzy number R=0.4, the level indicates the user might not have a trait; Otherwise Hi = 0.8-1, and the fuzzy number R=0.2, the level indicates the user don't have trait. The pseudo code for the algorithm is presented below:

```
$lastTime = 0;
while($data!=null)
{
        $d = $data[0];
        $data = $data[1:end];
        if($d['time']==lastTime){
                $Xm = $Xm+$d;//put current data into Xm
        }
        else{//current Xm is full and the new data should be add to a
new Xm
                $lastTime = %d['time'];
                        $uncert = 0.75;
                $assessScore = covariance of ($Xm)
if($assessScore>=0.8){//Fuzz level 5
                        Table-background-color:Red;
                }
                else if($assessScore>=0.6){//Fuzz level 4
                        Table-background-color:Orange;
                }
                else if($assessScore>=0.4){//Fuzz level 3
                        Table-background-color:Yellow;
                }
                else if($assessScore>=0.2){//Fuzz level 2
                        Table-background-color:Green;
                }
                else{//Fuzz level 1
                        Table-background-color:Blue;
                }
                $Xm = $newXm;
        }
}
```

In GUI, we display a ReadingBehaviorObservation table with each row as a segment (Figure 18). The background color shows the trait level when user is reading within this one-second segment.



0.72193868536029

Xm	# of Xi	Assessment Score1	Assessment Score2	Assessment Score Avg.	FuzzNum	Uncertainty
1	4	0.14890001959662	0.1318845367093	0.14039477815306	0.71821044389366	0.75
2	6	0.18819593184204	0.0880789900144336	0.13713745589319	0.72572508821362	0.75
3	3	0.48658124640194	0.1960834260109	0.34133233620642	0.31733532756715	0.75
4	15	0.17364544144138	0.18327857731016	0.17846100937576	0.84307798124847	0.75
5	18	0.13328470359942	0.15567282003287	0.15946678181614	0.68106247636771	0.75
6	13	0.23334382963704	0.16885631531977	0.20110007247784	0.59779965504319	0.75
7	4	0.30436181911267	0.32599923672092	0.3151805269166	0.36963894616641	0.75
8	4	0.2904531206578	0.29647019598103	0.29346185831941	0.41307966336117	0.75
9	12	0.15989906780798	0.12713189667726	0.14356035384039	0.71287923231921	0.75
10	12	0.29935450196887	0.27424095117043	0.28679772656965	0.4264045468607	0.75
11	14	0.27443905582572	0.28715247416796	0.27079578501184	0.45840846997632	0.75
12	11	0.24169166262784	0.16968923228017	0.2056804574439	0.58863909011219	0.75
13	15	0.33456234136029	0.34472371990497	0.33964303063263	0.32071393873474	0.75
14	15	0.19984612353587	0.14173584184958	0.17079098274278	0.65841803451443	0.75
15	14	0.14937376083627	0.25707366096665	0.20322372090156	0.59355255819688	0.75
16	9	0.32754454469131	0.17768394679747	0.25261424674439	0.49477150651122	0.75

Figure 18. Reading Behavior Observation Table.

Besides fuzzy number, we also include uncertainty number in our System to indicate the capability of our prediction. The uncertainty number for random guess is 0.5. We initially set it as 0.75 for our system. This uncertainty number will be updated based on future user study prediction accuracy.

We include a HeatMap, which is a saliency map of user's attention, as an overview (Figure 19).

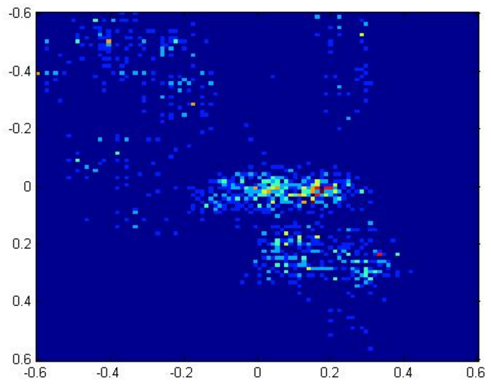


Figure 19. HeatMap of Reading Indicating High Meditation Trait in user's reading of CHI manual.

6. Related Work

Mindfulness meditation along with relaxation and biofeedback are major self-regulatory strategies that are wildly explored in clinical used therapy [3]. Mindfulness meditation has been proven to have positive effects on social skills, feeling of compassion, self-management, somatic awareness [7] and mental flexibility [8]. Besides that, studies has been work on usage of mindfulness meditation in treatment of anxiety disorders, stress reduction [3], [5], chronic pain and persistent pain [5], [9], depression [5], autism spectrum disorders [5], traumatic experiences [10], acquired brain injury [11], and even disordered eating, psoriasis and substance abuse [12], [13] and so on. Concentrative meditation, based on its unlimited physical form, is being taught broadly for stress reduction. However, the traditional researches on mindfulness as well as concentrative meditation mostly relied on self-report and verbal comprehension as measurements [3], [4], [5].

Transcendental meditation is a technique that turns "the attention inwards towards the subtler levels of a thought until the mind transcends the experience of the subtlest state of the thought and arrives at the source of the thought" [15], which can be classified as concentrative meditation [2]. Because of the easiness and enjoyableness, large subject number, repetition of mantra [2], and immediately experience beneficial physiological changes [16], TM becomes a popular meditation technique that being measured via physiological signals [[2], [6], [14]. Metric of meditation measurement via physiological signals consists of two major parts: state and trait. State is "altered sensory, cognitive, and self-referential awareness that can arise during meditation practice" and trait is "the lasting changes in these dimensions that persist in the meditator irrespective of being actively engaged in meditation" [2].

Among all physiological signals being used to measure TM, brain signals via electroencephalography (EEG) have more than 60 years history and are most commonly used [2]. Hans Berger first recorded EEG signals in the 1920s [20]. After about 90 years development, EEG now can be divided into 6 bands by frequency: alpha, beta, gamma, theta, delta, and mu. Alpha and theta bands are highly related to meditation state [18], [21], [22], [23], [24], [25]. Although many researchers [21], [22], [23] found that alpha power increases during meditating, different results have been reported based on different location of EEG sensors (i.e. frontal, parietal, temporal, or occipital) [18]. Besides EEG, physiological signals of ERP [2], GSR [6], Oxygen consumption [6], HRV [6] and neuroimaging [2] have been applied to monitoring meditation.

Our TDR-CHI Gaze component was inspired by [17], which tracks the gaze duration of a subject in meditation, and control-subjects on different emotional face stimuli and found that meditators spent less time on angry and fear faces than control subjects. TDR-CHI Gaze component has two functions: 1) State-Meditating: measuring state during meditating via face-tracking and eye-tracking technique, 2) Trait-Reading: evaluating trait during reading. Gaze component is different from previous works in three aspects: A. We creatively use face tracking and eye tracking technique to monitor meditation state (State-Meditating). To our knowledge, we are the first one that attempts to use face tracking and eye-tracking technique to track appearance changes in order to understand the internal changes in meditating. B. We track users reading patterns to determine users' binary meditation trait. We are changing the simple stimuli task to complicated reading task. C. The state and trait are monitored via smart phone, without dedicated wearable sensors and eye tracking sensors.

7. Discussion

We proposed in Section 7 to determine medication state by analyzing gaze obtained from the smart phone's camera. The initial experimental results indicate the approach might be viable. More experiments are to be performed.

If the brain wave headset is to be used, we would like to use the least amount of data to train a model, so that users do not need to get a device to monitor 16 channels. A smaller and cheaper device that only monitor a few channels may be good enough for meditation status prediction. To select few channels, we used information gain for feature selection. Since the records are mostly plain data from each channel, thus, by using feature selection technique, it can select channels that can differentiate data most efficiently.

Another solution is through Principal Component Analysis (PCA). With help of PCA, the system can find out which feature (channel) contributes most to the data. Then we may use one or more most contributes channels to be our final channels. This will lead to the most efficient (fewest channel) headset.

Our long-term goal is to expand the TDR system for the estimation of Chi. The Chi super-component can be regarded as the super-component at the highest level. It has attributes including both objective measurements and subjective evaluations. Some researchers propose to employ electrical measurements to estimate Chi [26]. Other researchers propose to combine objective measurements with subjective evaluation into an evaluation matrix to estimate Chi [27]. This makes the Chi super-component both pro-active and adaptive at multiple levels.

Finally we also want to add audio output so that a person's health exercise can be further enhanced. Our grand hypothesis is the mobile TDR system with multiple sensors will facilitate the estimation of Chi, so that a person equipped with the mobile TDR system can practice meditation and continue to enhance his/her health. Further experimental research will hopefully confirm at least a portion of this grand hypothesis.

Acknowledgement

ZhenJiang Fan contributed to the implementation of mobile TDR system.

References

[1] Shi-Kuo Chang, JunHui Chen, Wei Guo and Qui Zhang, "TDR System - A Multi-Level Slow Intelligence System for Personal Health Care", Proceedings of 2016 International Conference on Software Engineering and Knowledge Engineering (SEKE2016), Hotel Sofitel, Redwood City, San Francisco Bay, California, USA, July 1-3, 2016, 183-190.
[2] Cahn, B. Rael, and John Polich. "Meditation states and traits: EEG, ERP, and neuroimaging studies." Psychological bulletin 132.2 (2006): 180.
[3] Peterson, Linda Gay, and Lori Pbert. "Effectiveness of a meditation-based stress reduction program in the reatment of anxiety disorders." Am J Psychiatry 149.7 (1992): 936-943.
[4] Kabat-Zinn, Jon, Leslie Lipworth, and Robert Burney. "The clinical use of mindfulness meditation for the self-regulation of chronic pain." Journal of behavioral medicine 8.2 (1985): 163-190.
[5] Spek, Annelies A., Nadia C. Van Ham, and Ivan Nyklíček. "Mindfulness-based therapy in adults with an autism spectrum disorder: a randomized controlled trial." Research in developmental disabilities 34.1 (2013): 246-253.

[6] Wallace, Robert Keith. "Physiological effects of transcendental meditation." Science 167.3926 (1970): 1751-1754.
[7] Reid, Denise T. "Teaching mindfulness to occupational therapy students: Pilot evaluation of an online curriculum." Canadian journal of occupational therapy 80.1 (2013): 42-48.
[8] Thompson, Barbara. "Mindfulness-based stress reduction for people with chronic conditions." The British Journal of Occupational Therapy 72.9 (2009): 405-410.
[9] Stroh-Gingrich, Bethany. "Occupational therapy and mindfulness meditation: An intervention for persistent pain." Occupational Therapy Now 14.5 (2012): 21-22.
[10] Ruff, Kelley McCabe, and Elizabeth R. Mackenzie. "The role of mindfulness in healthcare reform: a policy paper." Explore: The Journal of Science and Healing 5.6 (2009): 313-323.
[11] MFKF, Michel Bedard Dwight Mazmanian, Carrie Gibbons Gary Mack, and Rupert Klein. "A Mindfulness-Based Intervention to Improve Quality of Life Among Individuals Who Sustained Traumatic Brain Injuries: One-Year Follow-Up." The Journal of Cognitive Rehabilitation (2005).
[12] Kabat-Zinn, Jon, and Thich Nhat Hanh. Full catastrophe living: Using the wisdom of your body and mind to face stress, pain, and illness. Delta, 2009.
[13] Smalley, Susan L., and Diana Winston. Fully present: The science, art, and practice of mindfulness. Da Capo Press, 2010.
[14] Elliott, Nina. "Exploring mindfulness meditation in occupational therapy: An introduction to basic practice." Occupational Therapy Now 17.1 (2015): 6-8.
[15] Mahesh Yogi, Maharishi. Maharishi Mahesh Yogi on the Bhagavad-Gita: A new translation and commentary with Sanskrit text, Chapters 1 to 6. Harmondsworth: Penguin Books, 1969.
[16] Yogi, Mahesh. "The science of being and art of living." (1963).
[17] Pavlov, S. V., et al. "Effects of long-term meditation practice on attentional biases towards emotional faces: An eye-tracking study." Cognition and Emotion 29.5 (2015): 807-815.
[18] Jian-Zhou, Zhang, Li Jing-Zhen, and He Qing-Nian. "Statistical brain topographic mapping analysis for EEGs recorded during Qi Gong state." International Journal of Neuroscience 38.3-4 (1988): 415-425.
[19] Fernando Lopes da Silva. "EEG: Origin and Measurement. " URL= http://webcache.googleusercontent.com/search?q=cache:Cx4xXMFDePsJ:www.springer.com/cda/content/document/cda_downloaddocument/9783540879183-c1.pdf%3FSGWID%3D0-0-45-883705-p173899505+&cd=3&hl=en&ct=clnk&gl=us
[20] Berger, Hans. "Über das elektrenkephalogramm des menschen." European Archives of Psychiatry and Clinical Neuroscience 87.1 (1929): 527-570.
[21] Aftanas, L. I., and S. A. Golocheikine. "Human anterior and frontal midline theta and lower alpha reflect emotionally positive state and internalized attention: high-resolution EEG investigation of meditation." Neuroscience letters 310.1 (2001): 57-60.
[22] Echenhofer, F. G., M. M. Coombs, and L. Samten. "EEG and P300 differences during meditation and rest in advanced Tibetan Buddhist and beginning meditators." meeting of the Society for Psychophysical Research, San Diego, CA. 1992.
[23] Wallace, Robert Keith. "Physiological effects of transcendental meditation." Science 167.3926 (1970): 1751-1754.
[24] Fenwick, P. B. C., et al. "Metabolic and EEG changes during transcendental meditation: an explanation." Biological Psychology 5.2 (1977): 101-118.
[25] Travis, Fred, et al. "Patterns of EEG coherence, power, and contingent negative variation characterize the integration of transcendental and waking states." Biological psychology 61.3 (2002): 293-319.

[26] Ming-Feng Chen, Hsi-Ming Yu, Shu-Fang Li and Ta-Jung You, "A Complementary Method for Detecting Qi Vacuity", *BMC complementary and alternative medicine*, Vol. 9, No. 12, 2009.

[27] Ke-Feng Huang, *Effects of Energy Absorption on Meridian System* (能量攝取對經絡系統影響之效應), Doctoral Dissertation (in Chinese), Institute of Biomedical Engineering, National Yang-Ming University, Taiwan, June 2011.

Car2Car framework based on DDGP3

Walter Balzano, Vinicio Barbieri, Giovanni Riccardi
Dip. Ing. Elettrica e Tecnologie dell'Informazione
Università di Napoli, Federico II
Napoli, Italy
e-mail: wbalzano@unina.it, vinicio.barbieri@gmail.com, ing.giovanni.riccardi@gmail.com

Abstract— **The purpose of this paper is to provide an algorithm for the detection of free parking stalls within a multilevel garage. Obviously, we are in the condition where the parking is very busy. Using devices of the cars On Board Units (OBU) and Road Side Units (RSU) is possible to determine, with a certain error, the matrix of distances between all sensors. The way of information exchange between cars is the VANETs. Starting from the known position of the RSU and the mutual distance between all adjacent cars OBU is possible to obtain the position of all cars applying a Distance Geometry Problem (DGP) algorithm schema. Unfortunately, the complexity of these algorithms is NP-hard. Under some conditions, the DGP algorithm can switch from continuous to discrete and it can be solved with a sort of branch and pruning algorithm. We are interested in a DDGP3 that is a Discretizable Distance Geometry Problem in R^3 variant to be used as a starting point for our work. The resolution of the algorithm is equivalent to the resolution of a problem of intersection between three spheres. This problem is non-linear and, in some conditions, it is possible to obtain an approximate solution with linear techniques.**

Keywords – DGP; DDGP; V2V; WiFi positioning; car parking.

I. Introduction

The DGP consists in seeking the coordinates of a set of points (vertices) in three-dimensional space starting from the distances between them.

Let us denote by $G = (V, E, d)$ a weighted graph, where each vertex in V content corresponds to a point in space and there is an edge between two vertices if and only if the distance between them is known. The graph G represents a DGP type problem, which in turn is the problem of finding a function

$$x : V \to R^3 \qquad (1)$$

such that for each arc belonging to E (and thus for every pair of vertices u, v connected by an arch) it is true that

$$\|x(u) - x(v)\| = d_{uv} \qquad (2)$$

In its basic form this is a "constraint satisfaction problem" the solution of which can be represented as follows:

$$X = \{x_v : v \in V\} \qquad (3)$$

II. Related Works

The different approach used in [2] and [3] for the solution of this problem is to treat it as a problem of continuous global optimization in which the set of constraints is replaced by an error function *Largest Distance Error* (LDE) that measures the difference between the calculated distance and the one known:

$$LDE(\{x_1, x_2, ..., x_n\}) = \frac{1}{m} \sum_{\{u,v\}} \frac{\left| \|x_u - x_v\| - d_{uv} \right|}{d_{uv}} \qquad (4)$$

where *m* is the number of known distances.

The DGP solution can be obtained by minimizing this function, which is not convex and contains many local minima. One of the approaches used to solve the problem is to approximate the function using a sequence of uniformly convergent functions; therefore a collection *X* is a solution if and only if the *LDE* error function is 0.

The DGP is applied for the solution of problems of location in wireless networks in which you know the distance between sensors (in our case OBU), but do not know their location, except that for some fixed named anchor (in our case RSU), then used to solve the problem.

One area in which the DGP is heavily used is Biology "Molecular Distance Geometry Problem" (MDGP).

The DGP can be treated as discrete as long as certain conditions are respected, and in particular, given a graph $G = (V, E, d)$ and a *Total Order of all vertices*, we must consider the following two axioms*:*

1. We assume that $(1,2,3) \subset V$, they must be a "clique" (fully reachable graph) and $\forall i \in V : i > 3$ must occur that these 3 arcs $\{(i-3,i),(i-2,i),(i-1,i)\} \subset E$
2. $\forall i \in V : i > 2$ It must apply strictly the triangle inequality $d_{i-2,i} < d_{i-2,i-1} + d_{i-1,i}$

If these conditions are verified then the cosine of the angle of torsion of each quadruple of consecutive vertices can be calculated

10.18293/DMSVLSS2017-018

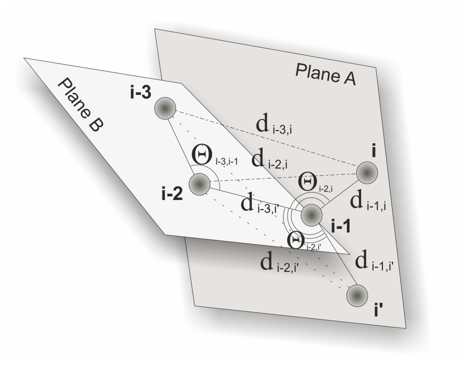


Figure 1: Angles of torsion of quadruple of consecutive vertices (through the intersection of two plans)

A position can be calculated for each one of two corners.

If these assumptions are verified the two possible positions can be calculated as the intersection of three spheres

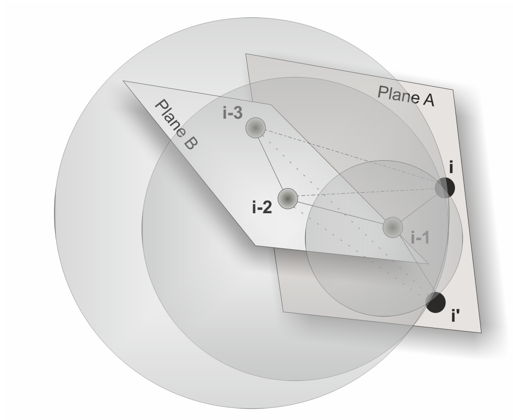


Figure 2: The intersection of the three spheres with center i-3, 1-2, i-1

- S_{-1} is the sphere with center in x_{i-1} and radius $d_{i-1,i}$
- S_{-2} is the sphere with center in x_{i-2} and radius $d_{i-2,i}$
- S_{-3} is the sphere with center in x_{i-3} and radius $d_{i-3,i}$

The intersection of the three spheres can be:

1. one point
2. two points
3. a circle
4. empty

The first hypothesis has probability 0, the third hypothesis is impossible (because of the strict triangular inequalities) and the fourth hypothesis is impossible (because the parking is very busy).

The only possibility, therefore, would be the second.

When these two assumptions are verified the LDE error function can be reduced to a discrete set and solved by the algorithm BP.

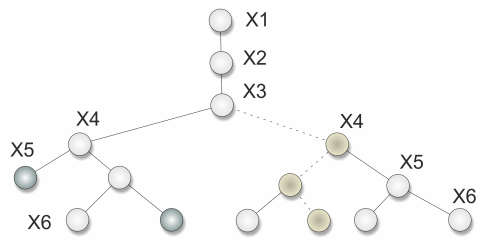


Figure 3: The algorithm BP branch and pruning

To switch from continuous to discrete domain this algorithm must be based on a real instance and these conditions must be verified:

- All distances required for the discretization (Axiom 1) are obtained from the OBU and RSU, so they are independent from the instance and stored in the VANETs;
- Distances between pairs (i, i + 1) and (i, i + 2) are know;
- Distances between pairs (i, i + 3) may be represented by intervals:
    1. $d_{i,i+3}$ is 0: it means that this is a duplicated car position, there is no branching because it can only take the same position of its previous copy;
    2. $d_{i,i+3}$ is exact: the standard discretization process is applied, and hence two possible positions for the current car position are computed;
    3. $d_{i,i+3}$ is represented by an interval: D sample distances are taken from the interval and the discretization process is applied for any chosen sample distance; 2×D car positions are generated.

III. C2C Framework

Our goal is to provide a complete procedure to find out a map highlighting the free stalls in a congested multilevel parking.

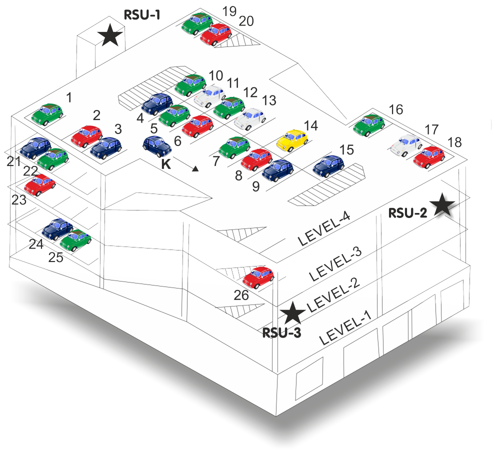


Figure 4: Parking Scheme with RSU and OBU

To achieve it we make the following assumptions:

1) *All vehicles are equipped with a sensor (OBU)*;
2) *Since each point of observation there are at least 3 fixed sensors (RSU), whose coordinates are known*;
3) *It is known the structure of the car park and are known the coordinates of all the parking stalls.*

The following flow chart shows the procedures used to obtain the map of free stalls. At each step the main input data are passed and the processing results constitute the input for the next step.

First of all, the algorithm load the map of parking and the position of Road Side Units (RSU), after that the set of vehicles registered on VANETs are loaded on memory.

All the information related to the mutual distance between the car are shared, but the On Board Units (OBU) load just the nearest.

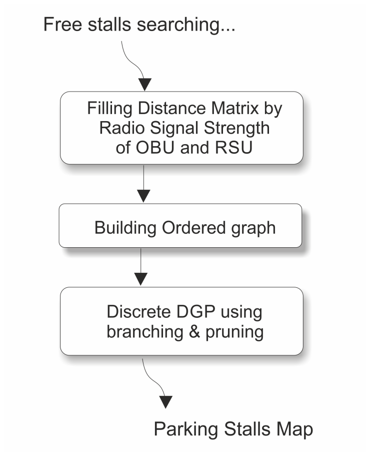

Free stalls searching...
Filling Distance Matrix by Radio Signal Strength of OBU and RSU
Building Ordered graph
Discrete DGP using branching & pruning
Parking Stalls Map

Figure 5: Workflow to search free stalls

A. Filling Distance Matrix by Radio Signal Strength of OBU and RSU

The model proposed in [1] allows proper accurate positioning where there are several vehicles in a small area, using a smart combination of RSS values transformed in a distance matrix provided by the V2V/V2I system.

To achieve the goal, we decide to use an external cloud[19] where to collect the distance vectors calculated by each vehicle. Such information is collected, and then these constitute the matrix of the distances which, by exploiting the cloud[20] computing capacity allows obtaining with a DGP algorithm map of the parking lot of vacancies.

Our proposal is not to use a cloud, but only assume a memory buffer on board each RSU able to accommodate distance carriers sent from the vehicles within a certain distance. Of course, knowing the structure of the parking lot and the location of the MSW is easy sizing the required memory.

Contrary to what was proposed in [1] we haven't the entire matrix of distances but only a part of it in each RSU.

When a vehicle enters the car park, in addition to calculating its distance vector, it requires RSU neighbours of distance carriers known to them and the map of the park, including the RSU coordinates and those of each parking stall.

The vehicle asks regularly update the data until it finds a parking stall and stops the vehicle. The map of the parking, being static information, once is transferred, as well as vectors of distances that do not change between a request and the next.

In this way, each vehicle has, at a certain instant, the partial matrix of the distances, the coordinates of the MSW and the parking map.

All this information will be processed by a DDGP algorithm to obtain the map of the parking lot with a list of free and occupied stalls.

B. Building Ordered graph

The Distance Geometry Problem (DGP) consists in finding the coordinates of a given set of points $\{x_1, x_2, \ldots, x_n\}$ in a three-dimensional space when some of the distances between pairs of such points are known. Our DGP instance is the set of vectors of distances received from neighboring RSU.

A vector of distances is considered significant if it contains more than three distances, sensors having less than three neighboring sensor could be initially removed from the network, and the localization problem may be solved for a sub-network (however, if a sensor has at least three sensors next it is likely that parking in that area is not congested and you do not need an algorithm to find a free place).

The graph to which we refer is G = (V, E, d), a weighted undirected graph associated to an instance of the DGP.

V = set of vertices, where each vertex in V corresponds to an c_i, in our case position of the vehicles and the RSU

E = set of arcs between vertices, there is an edge between two vertices only if it is known the distance between them (the weight associated to the edge)

d = set of distances between two vertices

We want to solve our problem by using a variant DDGP3 proposed in [3], this algorithm, depending on orders vertex and edge density.

To apply the DDGP3 algorithm it is necessary that the following condition is verified.

- (The three anchor sensors) {1, 2, 3} included in V are a "clique" (graph fully accessible), and for each parking stall occupied x_i belonging to V with rank i > 3, there are 3 vertices j, k , h such that:
  - $j < i,\ k < i, h < i,$
  - $(j, i), (k, i), (h, i) \in E$
  - $d_{jh} < d_{jk} + d_{kh}$.

To check the validity of this condition [3] suggests the following sorting algorithm:

High-level algorithm: Reordering Graph Vertices

```
Input:   $V_u$: Unordered vertices
Output:  $V_o$: Ordered vertices
 1: while( a valid ordering is not found ) do
 2:     find a 3-clique C in G($V_u$, E, d)
 3:     place the vertices of C at beginning of new
          order: G($V_o$, E, d) = C;
 4:     while($V_u - V_o \neq \emptyset$) do
 5:       find the vertex $v$ in $V_u - V_o$ with the largest number
                         of adjacent vertices in $V_o$;
 6:       if (l < 3) then
 7:           break the while loop: there are no possible
            ordering for this choice of C;
 8:       end if
 9:       $V_o = V_o + \{v\}$;
10:   end while
13: end while
14: return $V_o$
```

If the sorting algorithm found out a solution then we can move to the next step. We verified that the algorithm can find solutions if parking is congested (there are few free stalls).

C. Discrete DGP using Branching & pruning

Only after the sort, if the conditions are met, you can apply the DDGP3 algorithm. However, in order to avoid considering equivalent solutions that can be obtained from a given solution by translations or rotations, the first three points can be fixed. So that the final binary tree has 2n-3 positions. The first three positions are those of the RSUs closest to the viewer. At this point, we proceed to the calculation of the position and to the examination of the solution.

High-level algorithm: B&P

```
Input:   k, n, d.
Output:  Position of all vertices.
 1: for (i=1, 2) do
 2:     compute the $i^{th}$ position for the vertex k: $x_k^{(i)}$;
 3:     check the feasibility of the position $x_k^{(i)}$:
 4:     if (the position $x_k^{(i)}$ is feasible) then
 5:         if (k=n) then one solution is found;
 7:         else
 8:           It calls itself with these parameters (k+1, n, d)
 9:         end if
10:     else the current branch is pruned
12:     end if
13: end for
14: return Position of all vertices
```

The key points of the algorithm are two: the calculation of the intersection between the three spheres and check the feasibility of the 2 solutions found.

For the calculation of intersection points we are considering whether to use the algorithm proposed by [2] MD-jeep, or the more general technique proposed by [4]. Both approaches provide solution within a reasonable elaboration time.

What really makes a difference to the convergence of branch and prune algorithm is the feasibility check.

The idea is to exploit the condition 3), in fact, knowing the coordinates of the "center" of each stall, we can say that a point (intersection of three spheres) is acceptable if its coordinates fall in turn within a of spheres of radius R (R-value to be defined) which has the center coordinates (a priori known) of the stalls.

Assuming C = $\{(X_{c1}, Y_{c1}, Z_{c1}) .. (X_{cn}, Y_{cn}, Z_{cn})\}$ the set of coordinates of all “stall’s center”, the check to apply to each (X_x, Y_x, Z_x) coordinate is:

d $((X_x, Y_x, Z_x), (X_{ci}, Y_{ci}, Z_{ci})) < R$ for a (X_{ci}, Y_{ci}, Z_{ci}) in C

where:
d (A,B): distance from the points A and B
R: radius of the sphere to be fixed (i.e. 0.5 meters)

We chose the sphere only for simplicity of calculation, you can also think of a box (more realistic).

In addition we suggest eliminating the solutions that certainly do not make sense, such as those having the Z coordinate unacceptable. Only the points for which the z coordinate (height) is compatible with the heights of the various parking levels are acceptable.

Assuming H = $\{h_1 .. h_n\}$ the set of heights of the parking floors, the check to apply to each Z_x coordinate is:

$(h_i + h_{min}) < Z_x < (h_i + h_{max})$ for a h_i in H

where:
h_{min} : minimum height of a vehicle (i.e. 0.2 meters)
h_{max} : maximum height of a vehicle (i.e. 2 meters)

Knowing the map and the coordinate of each individual stall, the algorithm replaces the calculated value of coordinates with the nearest known one.

In this way, the result is much more precise and the application is more suitable.

IV. Experimental Results

In order to facilitate the implementation of the simulation procedure, used to test the algorithm, we have used a mathematical model [21; 22] under the following simplifying hypotheses:

1. We are considering uniform configurations of parking stalls on similar floors in order to simplify the simulation. Multi-level car park with 4 floors with 250 stalls per floor and.
2. In the real case the sensors are at a distance from the floor of the parking variable from a few centimeters up to a little more than one meter. In the simulations we assume to have all sensors exactly at the level of the membership plan.
3. A further simplification we did say we have the sensors in the center of each occupied stall.
4. The distance measurement even if in reality will be affected by the error simulations we considered accurate.

We have implemented the procedure in java and performed tests on a notebook with i5 processor and 8 GB of RAM and Ubuntu 17.04 operating system.

The problem consists in verifying if a stall is free or is occupied by a sensor (vehicle).

The procedure takes as input a matrix of distances which we assume it has been acquired by the sensors, also knowing the map and the coordinates of each single stall we have the possibility of replacing the calculated value of the stall coordinates with the known coordinates of the nearest stall (within a radius R) to improve the approximation of the calculated position up to reduce to zero the error

Below is a table summarizing the results obtained in four test. As will be noted, in the table we do not report the error columns because, at each step of problem resolution of the intersection of three spheres (on which is based the whole algorithm) we obtain the coordinates of two points and choose the one feasible with respect to the parking structure.

The algorithm identifies a point feasible substitute its coordinates with coordinates known a priori.

The tests are all performed in cases of high congestion of the parking lot, and as you can see the convergence of the times are quite stable and low, in fact, are between 0.11 and 0.19 seconds. These times are encouraging for us and make us think of a future implementation on mobile devices.

TABLE I. RESULTS OF CAR2CAR ALGORITHM

Tests	**Boundary Conditions**		
	Number of stalls	***Number of OBU per floor (occupied stalls)***	***Duration***
80% occupied stalls evenly on all floors	1000	200 - 200 - 200 - 200	0.11 sec
80% occupied stalls with higher density on the lower floors	1000	244 - 222 - 195 - 139	0.12 sec
90% occupied stalls evenly on all floors	1000	225 - 225 - 225 - 225	0.15 sec
90% stalls occupied with higher density on the lower floors	1000	249 - 242 - 226 - 183	0.19 sec

V. CONCLUSIONS AND FUTURE WORKS

We started from a real problem, the search for a free place within a multi-level parking lot congested. Taking advantage of the sensors of the RSUs fixed and mobile units OBUs we calculated the distances between them, resulting in a matrix of the partial distances that we used as DGP instance. Placing particular conditions we solved the problem by using a DDGP3 algorithm. Finally we used a feasibility check that allowed us to have a rapid convergence of the algorithm.

The network model is a wireless network and each node of type OBU is able to communicate directly with any other node of same type and with one of RSU type.

The elaborations are made directly on the OBU in-car, and then the results are putted in a shared area memory through the VANETs.

In the future, we will try to improve both the calculation of the distances and the DGP algorithm. Our goal is to make the entire procedure usable with the sensors and the ability of calculation of smart phones.

REFERENCES

[1] Walter Balzano, Fabio Vitale. "DiG-Park: a smart parking availability searching method using V2V/V2I and DGP-class problem." 31st International Conference on Advanced Information Networking and Applications Workshops 2017 - DOI 10.1109/WAINA.2017.104

[2] Mucherino, Antonio, Leo Liberti, and Carlile Lavor. "MD-jeep: an implementation of a branch and prune algorithm for distance geometry problems." International Congress on Mathematical Software. Springer Berlin Heidelberg, 2010.

[3] Mucherino, Antonio, Carlile Lavor, and Leo Liberti. "The discretizable distance geometry problem." Optimization Letters (2012): 1-16

[4] Coope, I. D. "Reliable computation of the points of intersection of n spheres in n." ANZIAM Journal 42 (2000): 461-477.

[5] Balzano, Walter, Maria Rosaria Del Sorbo, and Silvia Stranieri. "A logic framework for c2c network management." Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on. IEEE, 2016.

[6] Lavor, Carlile, et al. "Discretization orders for distance geometry problems." Optimization Letters 6.4 (2012): 783-796.

[7] Balzano, Walter, et al. "A Logic-based Clustering Approach for Cooperative Traffic Control Systems." International Conference on P2P, Parallel, Grid, Cloud and Internet Computing. Springer International Publishing, 2016.

[8] Balzano, Walter, Aniello Murano, and Fabio Vitale. "V2V-EN–Vehicle-2-Vehicle Elastic Network." Procedia Computer Science 98 (2016): 497-502.

[9] Lavor, Carlile, et al. "On a discretizable subclass of instances of the molecular distance geometry problem." Proceedings of the 2009 ACM symposium on Applied Computing. ACM, 2009.

[10] Abdelhamid, Sherin, Hossam S. Hassanein, and Glen Takahara. "Vehicle as a mobile sensor." Procedia Computer Science 34 (2014): 286-295.

[11] Sładkowski, Aleksander, and Wiesław Pamuła, eds. Intelligent Transportation Systems–Problems and Perspectives. Vol. 32. Springer, 2015.

[12] Balzano, Walter, Maria Rosaria Del Sorbo, and Domenico Del Prete. "SoCar: a Social car2car framework to refine routes information based on road events and GPS." Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on. IEEE, 2015.

10.18293/DMSVLSS2017-018

[13] Y. Allouche, M. Segal, “Cluster-based beaconing process for VANET”, Vehicular Communications Volume 2, Issue 2, April 2015, Pages 80–94.

[14] Allouche, Yair, and Michael Segal. "Cluster-based beaconing process for VANET." Vehicular Communications 2.2 (2015): 80-94.

[15] Huang, Chi-Fu, Yuan-Feng Chan, and Ren-Hung Hwang. "A Comprehensive Real-Time Traffic Map for Geographic Routing in VANETs." Applied Sciences 7.2 (2017): 129.

[16] Sanguesa, Julio A., et al. "RTAD: A real-time adaptive dissemination system for VANETs." Computer Communications 60 (2015): 53-70.

[17] Milojevic, Milos, and Veselin Rakocevic. "Distributed road traffic congestion quantification using cooperative VANETs." Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean. IEEE, 2014.

[18] Monteil, Julien, et al. "Distributed and centralized approaches for cooperative road traffic dynamics." Procedia-Social and Behavioral Sciences 48 (2012): 3198-3208.

[19] Amato, F., Moscato, F. Exploiting Cloud and Workflow Patterns for the Analysis of Composite Cloud Services (2017) Future Generation Computer Systems, 67, pp. 255-265. DOI: 10.1016/j.future.2016.06.035

[20] Amato, F., Moscato, F. Pattern-based orchestration and automatic verification of composite cloud services (2016) Computers and Electrical Engineering, 56, pp. 842-853. DOI: 10.1016/j.compeleceng.2016.08.006

[21] Amato, F., Moscato, F. Model transformations of MapReduce Design Patterns for automatic development and verification (2016) Journal of Parallel and Distributed Computing. DOI: 10.1016/j.jpdc.2016.12.017

[22] Amato, F., Moscato, F. A model driven approach to data privacy verification in e-health systems (2015) Transactions on Data Privacy, 8 (3), pp. 273-296.

[23] Oka, Hiroaki, and Hiroaki Higaki. "Multihop data message transmission with inter-vehicle communication and store-carry-forward in sparse vehicle Ad-hoc networks (VANET)." New Technologies, Mobility and Security, 2008. NTMS'08.. IEEE, 2008.

[24] Smith, David J. Reliability, maintainability and risk: Practical methods for engineers including reliability centred maintenance and safety-related systems. Elsevier, 2011.

[25] Li, Wenfeng, et al. "On reliability requirement for BSM broadcast for safety applications in DSRC system." Intelligent Vehicles Symposium Proceedings, 2014 IEEE. IEEE, 2014.

[26] Monteil, Julien, et al. "Distributed and centralized approaches for cooperative road traffic dynamics." Procedia-Social and Behavioral Sciences 48 (2012): 3198-3208.

[27] Sanguesa, Julio A., et al. "RTAD: A real-time adaptive dissemination system for VANETs." Computer Communications 60 (2015): 53-70.

Sentiment Analysis on Yelp social network

Flora Amato, Giovanni Cozzolino, Antonino Mazzeo and Antonio Pizzata
Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI
University of Naples "Federico II", ITALY
Email: {flora.amato, giovanni.cozzolino, mazzeo, antonio.pizzata}@unina.it

Abstract

Social networks analysis is an emerging trend among scholars and researchers in the last years. A great number of companies are interested in social networks data mining. Data gathered from Facebook, Twitter or other social networks result to be very attractive in many application fields, like economics analysis, sentiment analysis, and politics analysis and so on. In our work, we focused on the analysis of the content of textual information obtained from the social media. Our investigation is finalized to extract hot topics in social network. We considered, as case study, reviews obtained from the social network Yelp.

1 Introduction

Nowadays, social network information are precious for many application fields. For example information gathered from social network are used by so-called "social media marketing" to organize an advertising campaign: companies can reach a bigger number of stakeholders and create a more modern concept of advertising, quicker, immediate and sometimes matched with catchy hashtags.

Important information that analyst are interested in, regarding how people feel about an event occurred in their lives or around them. It has been proved [21] that the use of emoticon or hashtags can identify the sentiments people feel: the use of these forms of communications are the easiest way to share them on the net. This led researchers and scholars to extend the study of sentiment analysis[14, 15] to the digital world, mining information through forums, blogs, social media, in order to understand how the people react to and event (i.e. a new law, terrorist attack and so on).

In literature, there are many tools and projects designed to perform different kind of analysis[4].

In our work, we focused on the analysis of the content of textual information obtained from the social media: through our investigation, it is possible to extract hot topics in social network. We considered, as case study, reviews obtained from the social network Yelp.

1.1 Text Mining Procedures

Effective and efficient access to domain relevant information requires the ability to automatic process and organize the information especially if these are contained in huge repositories of data [7, 6]. The most used approaches in Big Data processing are based on the graph algorithms, parallel and distributed architecture.Some Big Data infrastructures deal with Apache Hadoop [26] software for data-intensive distributed applications, based in the MapReduce programming model and a Distributed File System (Hadoop). MapReduce job splits the input dataset into independent subsets that are dealt with map tasks in parallel. This step of mapping is then followed by a step of reducing tasks. These reduce tasks use the output of the maps to compute the result of the job. Some open source tools for Big Graph mining are proposed, as Pegasus, a big graph mining system built on top of MapReduce.

It allows to find patterns and anomalies in massive real-world graphs. Another Big Data Mining initiative is Apache a scalable machine learning and data mining open source software based mainly on Hadoop and a collection of hardware, software and design patterns for managing very fast large-scale data at very low cost and using BIDMat an interactive matrix library that integrates CPU and GPU acceleration.

For what concerns the text analysis, Morphosemantic approaches similar to the one proposed here have been already proposed for many languages and applied to the medical domain. Works that deserve to be mentioned are Pratt on the identification and on the transformation of terminal morphemes in the English medical dictionary; Wolff on the classification of the medical lexicon based on formative elements of Latin and Greek origin; Pacak et al. on the diseases words ending in -itis; Norton e Pacak on the surgical operation words ending in -ectomy or -stomy; Dujols et al.on the suffix -osis. Between the nineties and the 2000, many studies have been published on the automatic popula-

DOI reference number: 10.18293/DMSVLSS2017-004

tion of thesauri, we recollect among others Lovis et al.,that derives the meaning of the words from the morphemes that compose them; Lovis et al. that identifies ICD codes in diagnoses written in different languages; Hahn et al.that segments the subwords in order to recognise and extract medical documents; and Grabar e Zweigenbaumthat uses machine learning methods on the morphological data of the thesaurus SNOMED (French, Russian, English). Several works focused the problem to the definition and the implementation of a comprehensive architecture for information structuring, while the work is dedicated to resolve the issue of ensuring semantic interoperability of different entities by mapping the content of different corpora on a set of shared concepts.

For what concerns the decision support system in literature they are usually categorized in two typologies, Knowledge-based and non-Knowledge-based [13], [25]. The firsts are accurately described in [22]. AAPHelp, created in 1972, was an early attempt to implement automated reasoning under uncertainty. Other systems are Asbru, EON and PRODIGY [32]; PROforma, SAGE [31]; and the Clinical Reminder System [20]. The last one is based on the [29] Evidence-based medicine and provides evidence-based clinical guidelines. A more detailed and systematic overview on many other CDSS is described in [19]. Recently, many studies focused on medical information extraction from structured or unstructured texts. Fette [18] presents a IE systems that integrates medical unstructured information into a clinical data warehouse to transform into a structured format the information inserted by physicians in a clinical information system. Rink [28] proposes a method for the automatic extraction of medical concepts and relations from electronic medical reports. Medical concepts are extracted with supervised machine learning algorithms. Several knowledge sources are used for feature extraction: a semantic role labeller, a POS tagger, a phrase chunk parser, WordNet, Wikipedia and the General Inquirer lexicon. Doan [17] introduced an automated system to extract medications and related information from discharge summaries. The researchers developed an integrated system adapting some existing NLP tools. In order to proper model data, several approach have been proposed [23, 24, 5]. Moreover, to efficiently process huge amount of data, several approach regarding hardware implementation of data processing tools are developed[2, 3]. In particular, in[11] and [10] authors proposed an hardware implementation of a Decision Tree based multi-classification system traffic analyzer . The system is able to deal with a huge amount of data and tight constraints, such as power consumption and hardware resources. In[9] a traffic analysis hardware accelerator, based on the Decision Tree model, is presented through an infrastructure which collects data from mobile devices and provide them update versions of the analyzer by exploit new traffic information[8]. Moreover, in the field of data protection, in [12] authors proposed a secure infrastructure to protect intellectual property installed on the FPGA by means of partial dynamic configuration.

2 A Social Network Analysis Methodology

The predominant approach to analyze social network is the graph theory, even though it is largely debated. This theory derives from the studies of Euler and provides us a way for studying Networks of any kind. In social networks the single user or groups of users are represented as a point and their relation are represented as lines. The data obtained from these graphs are then recorded in matrix form, in this way we can study directly the data without drawing the graph, that helps a lot when we are facing large social network data sets. To the lines connecting points in the diagram we can assign a direction in order to determine which point influence the other and to that influence we can also assign a value to represent the strength of that relation.

2.1 Text Analysis Tools

2.1.1 TaLTac

TaLTaC (Trattamento automatico Lessicale e Testuale per lanalisi del Contenuto di un Corpus, Lexical and Textual automatic processing for analyzing the Content of a Corpus)[30] is a software able to perform on documents and data written in natural language operations like: Text Analysis, Text Mining and Corpus Analysis. It has been developed in Italy from the conjunction of the University of Salerno and University La Sapienza of Rome. The first task to do in our environment is creating a Work Session, which is the file that is going to contain all our information, then we build the Corpus, our main object, which will be analyzed with various instruments and operations. The Corpus is then divided in two parts: fragment and section. The first is identified by four asterisks (****) a name and some variables, if needed, identified themselves with an asterisk, a name and a value. The latter is defined with four plus signs (++++) and a name. Every Corpus can contain more fragments and sections. One of the main functions of TaLTaC is the extraction of significant information from the Corpus (Text Mining), such task uses endogenous and exogenous resources: the former is composed of the number of fragments and the categorical variables which can be associated to the text in order to identify fraction of the corpus logically related. Thanks to this kind of resource TaLTaC is able to perform a Specificity Analysis. The exogenous resources are lists which contain the frequency of a term or lexical unit, thanks to these lists the software is able to identify peculiar language of the text.

2.1.2 Gate

GATE (General Architecture for Text Engineering) [27] is a open source free software which excels at Text Analysis. The first version was written in the mid-1990s and it has reached currently version 8. The main resource of this program is ANNIE (A Nearly New Information Extraction System), which provides a lot of information extraction techniques such as the English Tokeniser which splits the text into annotations of type Token or the POS Tagger that assigns to every token an annotation that describes its characteristic (i.e NNP for Proper Noun in singular form) and more. The Gate software is a family made up of:

- an IDE, Gate Developer
- a web app, Gate Teamware
- a framework, Gate Embedded

2.2 The Methodology

In order to achieve a fine analysis of our data we applied different types of operations.

2.2.1 Text Pre-treatment

This is the first process to be applied on text, in order to obtain a clear analysis in the successive processes. It is made up of the following phases. **Normalization:** This phase allows to remove any data duplication and it normalizes the writings of names, acronyms and other entities. In order to achieve these goals we have to execute various tasks:

- Change apostrophes into stresses (for the words that is needed), in order to determinate the right word;
- Label words/sequence of words so that they can assume the right meaning and are not mistook with others expressions (i.e. a name can be mistaken for a noun, Rose or rose);
- Change of capital letters into lowercase for the words that are not labeled, if one is labeled we need to analyze what the label says: if it is a name (of person, of a city or a general proper name) then it will keep the capital letter, in other cases it will not.

Correcting Spelling Errors: This phase consists of comparing a misspelled word with the system dictionary in order to correct and analyze it in the right way.

2.2.2 Lexical Analysis

Lexical analysis analyzes the segments of the Corpus (a segment is a sequence of graphic forms separated by a strong divider). A segment can be easily be defined by choosing a set of dividers (i.e. punctuation such as .,;) and then separating the sentences between these elements. Once we obtained our segments we can estimate various analysis parameter such as the IS index: this measures the level of absorption of the segment regard the single elements which it is made of. Other important operations in this part of the analysis are the Tagging, which links to every word a description of the grammatical or semantic characteristics[16, 1], and the Lexation which identifies the sequence of words defined during the pretreatment as one unique entity. Last we define the Corpus key words by studying the repetition rates , the ones that have a noticeable standard deviation (considering only the integers) can be assumed being meaningful.

2.2.3 Textual Analysis

The first step in this analysis is the study of the Concordances in which we can examine the context where every word or segment we choose is. Then we calculate the TF-IDF rate which sorts the researchs results according to the frequency and distribution of the search keyword in the documents provided. The TF-IDF is equal tf-idf $= tf \cdot \log \frac{N}{n}$ where tf is number of occurrences of an element, and the remaining part is the logarithm of the ratio between the number of documents building the Corpus (N) and the number of documents which present that element (n). Another import part of Textual Analysis is the co-occurrences identification, where with co-occurrences we identify those couple of near elements that repeat in the text. This identification is useful to define the primary concepts contained in the Corpus.

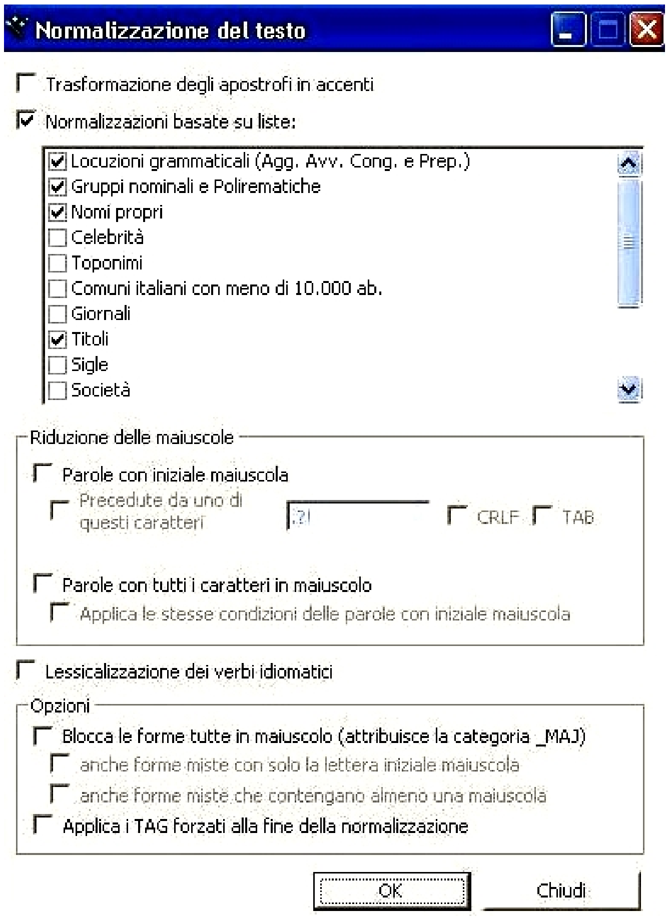
3 Experimental Campaign

Our experimentation aims to analyze online social networks data sets in order to derive as much information as possible. The data set analyzed comes from the social network Yelp, founded in 2004, which publishes crowd-sourced reviews about local businesses[33] and it is made up of 50 tuples structured this way:

- anonymized user name
- anonymized reviewed place
- date of the review
- review

3.1 TaLTac Analysis

Prior starting the analysis we pass through the Text Pretreatment, and so after the parsing of the Corpus we normalize (Figure 1) it and compute the sub-occurrences.



Normalizzazione del testo

- [ ] Trasformazione degli apostrofi in accenti
- [✓] Normalizzazioni basate su liste:
  - [✓] Locuzioni grammaticali (Agg. Avv. Cong. e Prep.)
  - [✓] Gruppi nominali e Polirematiche
  - [✓] Nomi propri
  - [ ] Celebrità
  - [ ] Toponimi
  - [ ] Comuni italiani con meno di 10.000 ab.
  - [ ] Giornali
  - [✓] Titoli
  - [ ] Sigle
  - [ ] Società

Riduzione delle maiuscole
- [ ] Parole con iniziale maiuscola
  - [ ] Precedute da uno di questi caratteri .?! [ ] CRLF [ ] TAB
- [ ] Parole con tutti i caratteri in maiuscolo
  - [ ] Applica le stesse condizioni delle parole con iniziale maiuscola

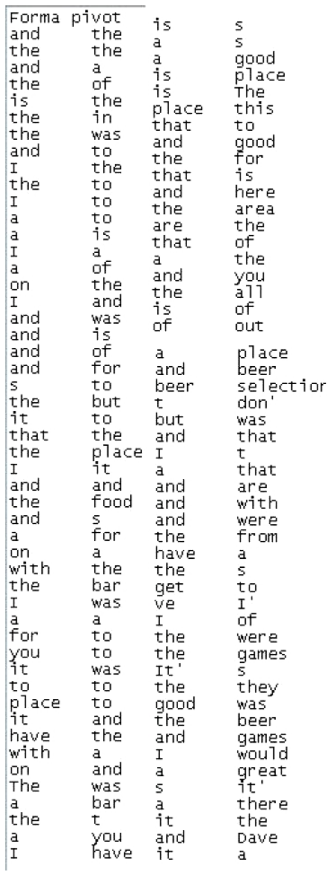
- [ ] Lessicalizzazione dei verbi idiomatici

Opzioni
- [ ] Blocca le forme tutte in maiuscolo (attribuisce la categoria _MAJ)
  - [ ] anche forme miste con solo la lettera iniziale maiuscola
  - [ ] anche forme miste che contengano almeno una maiuscola
- [ ] Applica i TAG forzati alla fine della normalizzazione

Figure 1. Text Normalization window

The next phase is the Textual Analysis, here we start with the identification of the segments, these are saved in two file: the former is Lista dei Segmenti (con indice IS) which contains the segments with their relative number of occurrences, number of elements forming the segment and the IS index; the latter, named Lista dei segmenti Significativi is a list of the significant segments. Then we analyze the specificity of our Corpus together with the computation of the TFIDF index, these data are saved in the same location under the name of: Vocabolario and Lessico. The last operation of our study is the Textual Analysis with the concordances and co-occurrences computation (Figure 2), the latter can be found in the file Cooccorrenze e collocazioni significative.

The results of these operations show that the system without knowing anything of the data submitted can retrieve meaningful information, such as the TFIDF index that shows how much a word is important in the document, or the co-occurrences which show the main concepts of the text thanks to the couple of words that recur all the time. Unfortunately TaLTac can not perform all the analysis on



```
Forma pivot
and     the     is      s
the     the     a       s
and     a       a       good
the     of      is      place
is      the     is      The
the     in      place   this
the     was     that    to
and     to      and     good
I       the     the     for
the     to      that    is
I       to      and     here
a       to      the     area
a       is      are     the
I       a       that    of
a       of      a       the
on      the     and     you
I       and     the     all
and     was     is      of
and     is      of      out
and     of      a       place
and     for     and     beer
s       to      beer    selectior
the     but     t       don'
it      to      but     was
that    the     and     that
the     place   I       t
I       it      a       that
and     and     and     are
the     food    and     with
and     s       and     were
a       for     the     from
on      a       have    a
with    the     the     s
the     bar     get     to
I       was     ve      I'
a       a       I       of
for     to      the     were
you     to      the     games
it      was     It'     s
to      to      the     they
place   to      good    was
it      and     the     beer
have    the     and     games
with    a       I       would
on      and     a       great
The     was     s       it'
a       bar     a       there
the     t       it      the
a       you     and     Dave
I       have    it      a
```

Figure 2. Co-occurences window

our data set due to the fact that it is in English and so our work is not totally complete. Even though our results are few, they are very significative.

3.2 GATE Analysis

Our first operation with this software is the initialization of ANNIE with Defaults, once all the Processing Resources are loaded we run ANNIE and start in sequence:

- Document Reset PR The document reset resource enables the document to be reset to its original state, by removing all the annotation sets and their contents;
- ANNIE English Tokeniser The tokeniser splits the text into very simple tokens such as numbers, punctuation and words of different types;
- ANNIE Gazetteer The role of the gazetteer is to identify entity names in the text based on lists;
- ANNIE Sentence Splitter The sentence splitter is a cascade of finite-state transducers which segments the text into sentences;

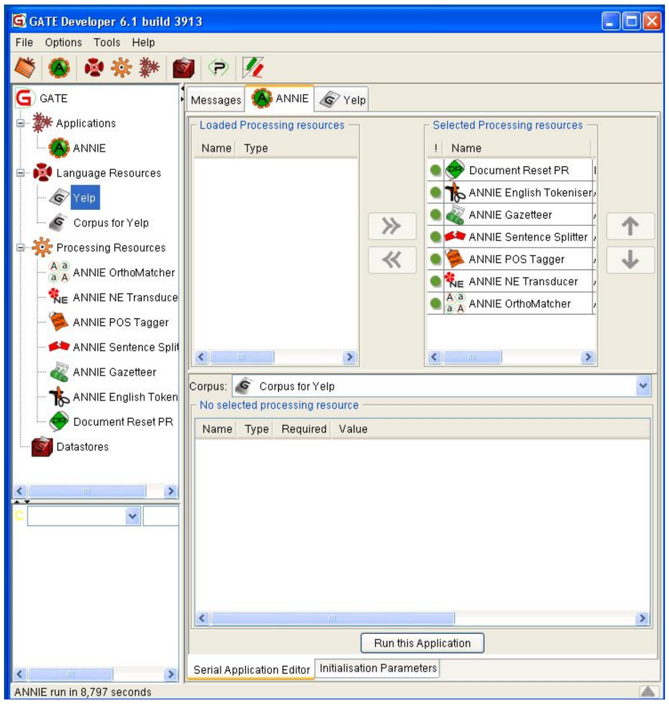


Figure 3. Annie Pipeline

- ANNIE POS Tagger The tagger produces a part-of-speech tag as an annotation on each word or symbol;
- ANNIE NE Transducer
- ANNIE OrthoMatcher The Orthomatcher module adds identity relations between named entities found by the semantic tagger, in order to perform coreference.

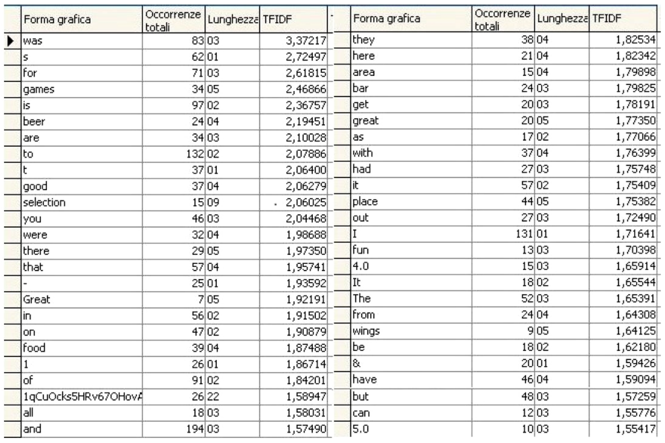
Thanks to these operations we can overcome TaLTac limits and run a Grammatical Tag on our data set. It must be said that our original data have been modified, indeed we are going to analyze only the review section due to the fact that in GATE the other information are meaningless.

4 Experimental Results

From our initial data set, made up of more than 5000 tuples, we extracted, as said, our 50 samples from which we obtained these results:

4.1 TaLTac results

Thanks to TaLTac functions we can say that our peculiar lexicon, shown in Figure 4, is composed by words with highest occurrences and TFIDF like: games, beer, good, great, food, place so it is reasonable to assume that the reviews analyzed talk about a place to eat food, drink beer or



Forma grafica	Occorrenze totali	Lunghezza	TFIDF	Forma grafica	Occorrenze totali	Lunghezza	TFIDF
was	83	03	3,37217	they	38	04	1,82534
s	62	01	2,72497	here	21	04	1,82342
for	71	03	2,61815	area	15	04	1,79898
games	34	05	2,46866	bar	24	03	1,79825
is	97	02	2,36757	get	20	03	1,78191
beer	24	04	2,19451	great	20	05	1,77350
are	34	03	2,10028	as	17	02	1,77066
to	132	02	2,07886	with	37	04	1,76399
t	37	01	2,06400	had	27	03	1,75748
good	37	04	2,06279	it	57	02	1,75409
selection	15	09	2,06025	place	44	05	1,75382
you	46	03	2,04468	out	27	03	1,72490
were	32	04	1,98688	I	131	01	1,71641
there	29	05	1,97350	fun	13	03	1,70398
that	57	04	1,95741	4.0	15	03	1,65914
-	25	01	1,93592	It	18	02	1,65544
Great	7	05	1,92191	The	52	03	1,65391
in	56	02	1,91502	from	24	04	1,64308
on	47	02	1,90879	wings	9	05	1,64125
food	39	04	1,87488	be	18	02	1,62180
1	26	01	1,86714	&	20	01	1,59426
of	91	02	1,84201	have	46	04	1,59094
1qCuOcks5HRv67OHovA	26	22	1,58947	but	48	03	1,57259
all	18	03	1,58031	can	12	03	1,55776
and	194	03	1,57490	5.0	10	03	1,55417

Figure 4. Peculiar lexicon by occurences

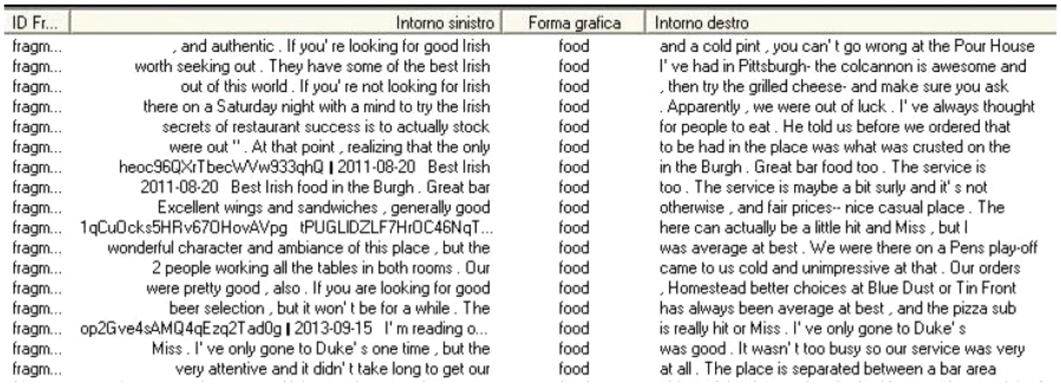
play some games and that the main audience thinks that it is a good place. Other meaningful data are the co-occurrences which are reported in Figure 2, these elements strengthen our thoughts about the place reviewed with expressions as: the bar, a good, a place, beer selection, the games. Last we took a meaningful word, food, and studied its concordances through all the Corpus (Figure 5), in order to understand the context of the lemma.

4.2 GATE results

Thanks to the Tokeniser we can distinguish spaces from words, and thanks to the Gazetteer and POS Tagger every word in our data set has a description. At the end of our analysis with this software we can assert the grammatical features of our data and we can give even more meaning to the analysis described in section 4.1.

5 Conclusions

Social networks analysis is an emerging trend among scholars and researchers in the last years. In literature, there are various instruments and project to achieve different kind of analysis, yet in our work, we focused on the analysis of the content of the text obtained from the social media. Through our investigation it was possible extracting the hot topics for the different information sources, as, in our case, the reviews analyzed. This study can be the starting point of further analysis on the domain of cybersecurity, through the detection of text containing dangerous messages, viral market advertising, thanks to an analysis of the feedback from the users or costumers, or information crawling.



ID Fr...	Intorno sinistro	Forma grafica	Intorno destro
fragm...	, and authentic . If you' re looking for good Irish	food	and a cold pint , you can' t go wrong at the Pour House
fragm...	worth seeking out . They have some of the best Irish	food	I' ve had in Pittsburgh- the colcannon is awesome and
fragm...	out of this world . If you' re not looking for Irish	food	, then try the grilled cheese- and make sure you ask
fragm...	there on a Saturday night with a mind to try the Irish	food	. Apparently , we were out of luck . I' ve always thought
fragm...	secrets of restaurant success is to actually stock	food	for people to eat . He told us before we ordered that
fragm...	were out '' . At that point , realizing that the only	food	to be had in the place was what was crusted on the
fragm...	heoc96QXrTbecWVw933qhQ ❙ 2011-08-20 Best Irish	food	in the Burgh . Great bar food too . The service is
fragm...	2011-08-20 Best Irish food in the Burgh . Great bar	food	too . The service is maybe a bit surly and it' s not
fragm...	Excellent wings and sandwiches , generally good	food	otherwise , and fair prices-- nice casual place . The
fragm...	1qCuOcks5HRv67OHovAVpg tPUGLIDZLF7HrOC46NqT...	food	here can actually be a little hit and Miss , but I
fragm...	wonderful character and ambiance of this place , but the	food	was average at best . We were there on a Pens play-off
fragm...	2 people working all the tables in both rooms . Our	food	came to us cold and unimpressive at that . Our orders
fragm...	were pretty good , also . If you are looking for good	food	, Homestead better choices at Blue Dust or Tin Front
fragm...	beer selection , but it won' t be for a while . The	food	has always been average at best , and the pizza sub
fragm...	op2Gve4sAMQ4qEzq2Tad0g ❙ 2013-09-15 I' m reading o...	food	is really hit or Miss . I' ve only gone to Duke' s
fragm...	Miss . I' ve only gone to Duke' s one time , but the	food	was good . It wasn' t too busy so our service was very
fragm...	very attentive and it didn' t take long to get our	food	at all . The place is separated between a bar area

Figure 5. Concordances of the word "food"

References

[1] M. Albanese, A. D'acierno, V. Moscato, F. Persia, and A. Picariello. Modeling recommendation as a social choice problem. pages 329–332, 2010.

[2] F. Amato, M. Barbareschi, V. Casola, and A. Mazzeo. An fpga-based smart classifier for decision support systems. *Studies in Computational Intelligence*, 511:289–299, 2014.

[3] F. Amato, M. Barbareschi, V. Casola, A. Mazzeo, and S. Romano. Towards automatic generation of hardware classifiers. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8286 LNCS(PART 2):125–132, 2013.

[4] Flora Amato, Antonino Mazzeo, Vincenzo Moscato, and Antonio Picariello. A framework for semantic interoperability over the cloud. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1259–1264. IEEE, 2013.

[5] R. Aversa, B. Di Martino, and F. Moscato. Critical systems verification in metamorp(h)osy. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8696 LNCS:119–129, 2014.

[6] Walter Balzano and Maria Rosaria Del Sorbo. Cotracks: A new lossy compression schema for tracking logs data based on multiparametric segmentation. In *Data Compression, Communications and Processing (CCP), 2011 First International Conference on*, pages 168–171. IEEE, 2011.

[7] Walter Balzano and Maria Rosaria Del Sorbo. Setra: A smart framework for gps trajectories' segmentation. In *Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on*, pages 362–368. IEEE, 2014.

[8] Walter Balzano and Fabio Vitale. Dig-park: a smart parking availability searching method using v2v/v2i and dgp-class problem. In *31st International Conference on Advanced Information Networking and Applications Workshops 2017 Computer Science*, 2017.

[9] Mario Barbareschi, Alessandra De Benedictis, Antonino Mazzeo, and Antonino Vespoli. Providing mobile traffic analysis as-a-service: Design of a service-based infrastructure to offer high-accuracy traffic classifiers based on hardware accelerators. *Journal of Digital Information Management*, 13(4):257, 2015.

[10] Mario Barbareschi, Salvatore Del Prete, Francesco Gargiulo, Antonino Mazzeo, and Carlo Sansone. Decision tree-based multiple classifier systems: An fpga perspective. In *International Workshop on Multiple Classifier Systems*, pages 194–205. Springer, 2015.

[11] Mario Barbareschi, Antonino Mazzeo, and Antonino Vespoli. Malicious traffic analysis on mobile devices: a hardware solution. *International Journal of Big Data Intelligence*, 2(2):117–126, 2015.

[12] Alessandro Cilardo, Mario Barbareschi, and Antonino Mazzeo. Secure distribution infrastructure for hardware digital contents. *IET Computers & Digital Techniques*, 8(6):300–310, 2014.

[13] Enrico Coiera. *Guide to health informatics*. CRC press, 2015.

[14] Francesco Colace, Massimo De Santo, and Luca Greco. A probabilistic approach to tweets' sentiment classification. In *Affective computing and intelligent interaction (ACII), 2013 humaine association conference on*, pages 37–42. IEEE, 2013.

[15] Francesco Colace, Massimo De Santo, Luca Greco, Vincenzo Moscato, and Antonio Picariello. A collaborative user-centered framework for recommending items in online social networks. *Computers in Human Behavior*, 51:694–704, 2015.

[16] A. D'Acierno, V. Moscato, F. Persia, A. Picariello, and A. Penta. iwin: A summarizer system based on a semantic analysis of web documents. pages 162–169, 2012.

[17] Son Doan, Lisa Bastarache, Sergio Klimkowski, Joshua C Denny, and Hua Xu. Integrating existing natural language processing tools for medication extraction from discharge summaries. *Journal of the American Medical Informatics Association*, 17(5):528–531, 2010.

[18] Georg Fette, Maximilian Ertl, Anja Wörner, Peter Kluegl, Stefan Störk, and Frank Puppe. Information extraction from unstructured electronic health records and integration into a data warehouse. In *GI-Jahrestagung*, pages 1237–1251, 2012.

[19] Amit X Garg, Neill KJ Adhikari, Heather McDonald, M Patricia Rosas-Arellano, PJ Devereaux, Joseph Beyene, Justina Sam, and R Brian Haynes. Effects of computerized clinical decision support systems on practitioner performance and patient outcomes: a systematic review. *Jama*, 293(10):1223–1238, 2005.

[20] U Kang, Duen Horng Chau, and Christos Faloutsos. Pegasus: Mining billion-scale graphs in the cloud. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 5341–5344. IEEE, 2012.

[21] Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[22] Randolph A Miller. Medical diagnostic decision support systemspast, present, and future. *Journal of the American Medical Informatics Association*, 1(1):8–27, 1994.

[23] F. Moscato. Model driven engineering and verification of composite cloud services in metamorp(h)osy. pages 635–640, 2014.

[24] F. Moscato. Exploiting model profiles in requirements verification of cloud systems. *International Journal of High Performance Computing and Networking*, 8(3):259–274, 2015.

[25] Mark A Musen, Blackford Middleton, and Robert A Greenes. Clinical decision-support systems. In *Biomedical informatics*, pages 643–674. Springer, 2014.

[26] The Apache Hadoop project. Apache hadoop.

[27] The GATE project team. Gate.

[28] Bryan Rink, Sanda Harabagiu, and Kirk Roberts. Automatic extraction of relations between medical concepts in clinical texts. *Journal of the American Medical Informatics Association*, 18(5):594–600, 2011.

[29] David L Sackett, William MC Rosenberg, JA Muir Gray, R Brian Haynes, and W Scott Richardson. Evidence based medicine: what it is and what it isn't, 1996.

[30] Adolfo Morrone Sergio Bolasco, Francesco Baiocchi. Taltac.

[31] Samson W Tu, James R Campbell, Julie Glasgow, Mark A Nyman, Robert McClure, James McClay, Craig Parker, Karen M Hrabak, David Berg, Tony Weida, et al. The sage guideline model: achievements and overview. *Journal of the American Medical Informatics Association*, 14(5):589–598, 2007.

[32] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The prodigy architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[33] The Free Encyclopedia Wikipedia. Yelp.

Sentiment Analysis on Yelp social network

Flora Amato*, Francesco Colace+, Giovanni Cozzolino*, Vincenzo Moscato*,
Antonio Picariello*, and Giancarlo Sperli*
*Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione DIETI.
University of Naples "Federico II", ITALY
Email: {flora.amato,giovanni.cozzolino, vmoscato,picus, giancarlo.sperli}@unina.it
+ Dipartimento di Ingegneria Industriale. DIIN. University of Salerno, ITALY
Email: fcolace@unisa.it

Abstract

In this paper, we propose a novel data model for Multimedia Social Networks, i.e. particular social media networks that combine information on users belonging to one or more social communities together with the content that is generated and used within the related environments. The proposed model relies on the hypergraph data structure to capture and represent in a simple way all the different kinds of relationships that are typical of social media networks, and in particular among users and multimedia content. We also introduce some user and multimedia ranking functions to enable different applications. Finally, some experiments concerning effectiveness of the approach for supporting relevant information retrieval activities are reported and discussed.

1 Introduction

Social media networks provide users an interactive platform to create and share multimedia content such as text, image, video, audio, and so on. Just as an example, each minute thousands of tweets are sent on Twitter, several hundreds of hours of videos are uploaded to YouTube, and a huge quantity of photos are shared on Instagram or uploaded to Flickr.

Within these "interest-based" networks, each user interacts with the others through a multimedia content and such interactions create "social links" that well characterize the behaviors of involved users in the networks. Here, multimedia data seems to play a "key-role" especially if we consider the *Social Network Analysis* (SNA) perspective: representing and understanding user-multimedia interaction mechanisms can be useful to predict user behavior, to model the evolution of multimedia content and social graphs, to design human-centric multimedia applications and services and so on. In particular, several research questions have to be addressed:

- It possible to exploit multimedia features and the notion of *similarity* to discover more useful links?
- Can all the different types of user annotations (e.g. tag, comment, review, etc.) and interactions with multimedia objects provide a further support for an advanced network analysis?
- Is it possible to integrate and efficiently manage in a unique network the information coming from different social media networks (for example, a Twitter user has usually an account also on Instagram or Flickr)?
- How can we deal with a very large volume of data?
- In this context, how is possible to model all the various relationships among users and multimedia objects[1]? Are the "graph-based" strategies still the most suitable solutions?

To capture the described issues, we adopt the term *Multimedia Social Networks* (MSNs) to indicate "*integrated social media networks that combine the information on users, belonging to one or more social communities, together with all the multimedia contents that can be generated and used within the related environments*".

Actually, the term MSN have been used over the last years in the literature together with *Social Multimedia Network* or *Social Media Network* to indicate information networks that leverage multimedia data in a social environment for several purposes: distributed resource allocation for multimedia content sharing in cloud-based systems [2], generation of personalized multimedia information recommendations in response to specific targets of interests [3],

DOI reference number: 10.18293/DMSVLSS2017-014

evaluation of the trust relationship among users [4], high dimensional video data distribution in social multimedia applications [5], characterization of user behavior and information propagation on the base of multimedia sharing activities [6], representation of a social collaboration network of archeologists for cultural heritage applications [7], just to cite some of the most recent proposals.

In this paper, inspired by hypergraph based approaches, we propose a novel data model[8, 9] for Multimedia Social Networks. Our model provides a solution for representing MSNs sufficiently general with respect to: i) a particular social information network, ii) the different kinds of entities, iii) the different types of relationships, iv) the different applications[10, 11]. Exploiting hypergraphs, the model allows us to represent in a simple way all the different kinds of relationships that are typical of a MSN (among multimedia contents, among users and multimedia content and among users themselves) and to enable several kinds of analytics and applications by means[12, 13] of the introduction of some user and multimedia (global and "topic sensitive") *ranking* functions.

We exploit functionalities of a well know framework for NLP processing, GATE [14] in order to extract relevant information from the famous online social network Yelp.

The paper is organized as in the following. Section 2 describes in details and using different examples our model with its properties and foundations. Section 3 shows the obtained experimental results using a standard Yelp dataset, while Section 4 reports conclusions and the future work.

2 The MSN data model

2.1 Basic Concepts

In our vision, a MSN is basically composed by three different entities:

- **Users** - the set of persons and organizations constituting the particular social community[15, 16]. Several information concerning their profile, interests, preferences, etc. can be exploited by our model.
- **Multimedia Objects** - the set of multimedia resources (i.e. images, video, audio, etc.) that can be shared within a MSN community. High level (*metadata*) and low level information (*features*) can be properly used in our model.
- **Annotation Assests** - the most significant terms or named entities - whose definition can be retrieved from dictionaries, ontologies and so on - of a given domain, or *topics*, exploited by users to annotate multimedia data and derived from the analysis of textual information such as keywords, labels, tags, comments etc.

Several types of relationships can be established among the described entities: for example, a user can annotate an image with a particular tag, two friends can comment the same post, a user can tag another user in a photo, a user can share some videos within a group and so on.

Due to the variety and complexity of these relationships, we decided to leverage the *hypergraph* formalism to model a MSN. In particular, our model relies on several concepts, *Multimedia Social Network* (seen as particular a weighted *hypergraph*) and *social paths* (i.e. *hyperpaths*), which basic definitions are provided in the following.

Definition 2.1 (MSN) *A* Multimedia Social Network *MSN is a triple $(V; H_e = \{e_i : i \in I\}; \omega)$, V being a finite set of* vertices*, H_e a set of* hyperedges *with a finite set of indexes I and $\omega : H_e \rightarrow [0,1]$ a weight function. The set of vertices is defined as $V = U \cup M \cup A$,* U *being the set of MSN users,* M *the set of multimedia objects and* A *the set of annotation assets. Each hyperedge $e_i \in H_e$ is in turn defined by a ordered pair $e_i = (e_i^+ = (V_{e_i}^+, i); e_i^- = (i, V_{e_i}^-))$. The element e_i^+ is called the* tail *of the hyperarc e_i whereas e_i^- is its* head*, $V_{e_i}^+ \subseteq V$ being the set of vertices of e_i^+, $V_{e_i}^- \subseteq V$ the set of vertices of e_i^- and $V_{e_i} = V_{e_i}^+ \cup V_{e_i}^-$ the subset of vertices constituting the whole hyperedge.*

Actually, vertices and hyperedges are *abstract data types* with a set of properties (attributes and methods) that permit to support several applications. We use the "dot notation" to identify the attributes of a given vertex or hyperedge: as an example, e_i.id, e_i.name, e_i.time and e_i.type represent the id, name, timestamp and type of the hyperedge e_i, respectively.

In addition, the weight function can be used to define the confidence or uncertainty of a given relationship in terms of probability, fuzzy membership, etc.

Definition 2.2 (Social path) *A* social path *between vertices v_{s_1} and v_{s_k} of a MSN is a sequence of distinct vertices and hyperedges $v_{s_1}, e_{s_1}, v_{s_2}, ..., e_{s_{k-1}}, v_{s_k}$ such that $\{v_{s_i}, v_{s_{i+1}}\} \subseteq V_{e_{s_i}}$ for $1 \leq i \leq k-1$. The* length *of the hyperpath is $\alpha \cdot \sum_{i=1}^{k-1} \cdot \frac{1}{\omega(e_{s_i})}$, α being a normalizing factor. We say that a social path* contains *a vertex v_h if $\exists e_{s_i} : v_h \in e_{s_i}$.*

Social paths between two nodes leverage the different kinds of relationships (see Section 3.2): a given path can "directly" connect two users because they are "friends" or members of the same group, or "indirectly", as they have shared the same picture or commented the same video.

2.2 Relationships

Analyzing the different types of relationships that can be established in the main social media networks, we have identified three categories:

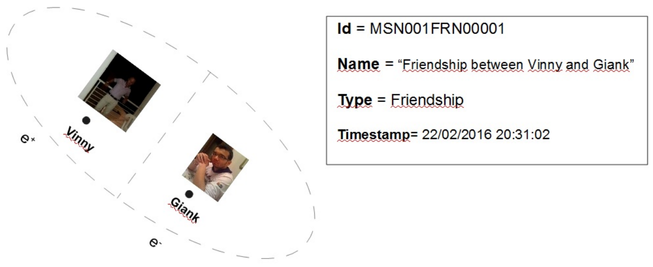
- **User to User** relationships, describing user actions towards other users;
- **Similarity** relationships, describing a relatedness between two multimedia objects, users or annotation assets;
- **User to Multimedia** relationships, describing user actions on multimedia objects, eventually involving some annotation assets or other users.

Definition 2.3 (User to User relationship) *Let $\widehat{U} \subseteq U$ a subset of users in a MSN, we define* user to user relationship *a hyperedge e_i with the following properties:*

1. $V^+_{e_i} = u_k$ *such that* $u_k \in \widehat{U}$,
2. $V^-_{e_i} = \widehat{U} - u_k$.

The weight function for such relationship returns as value $\frac{\hat{H}_{kj}}{H_k}$, $\hat{H}_{kj}$ being the average number of distinct user to user social paths between u_k and u_j for each $u_j \in \widehat{U} - u_k$, and H_k the number of user to user paths having as initial vertex u_k[1].

Examples of "user to user" relationships are represented by *friendship*, *following* or *membership* in On-line Social Networks[17]. To better explain this type of relationships, we provide in Figure 1 an example of *friendship* relationship.



Id = MSN001FRN00001

Name = "Friendship between Vinny and Giank"

Type = Friendship

Timestamp= 22/02/2016 20:31:02

Figure 1. Friendship relationship

Definition 2.4 (Similarity relationship) *Let $v_k, v_j \in V$ ($k \neq j$) two vertices of the same type of a MSN, we define* similarity relationship *a hyperedge e_i with the following properties:*

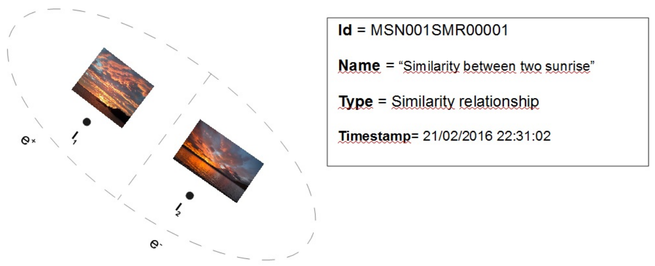
1. $V^+_{e_i} = v_k$,
2. $V^-_{e_i} = v_j$.

The weight function for this relationship returns similarity value between the two vertices.

The similarity relationships are defined on the top of a *similarity function* $f_{sim} : V \times V \rightarrow R$. It is possible to compute a similarity value:

- between two users by considering different types of features (interests, profile information, preferences, etc.);
- between two multimedia objects using the well-known (high and low level) features and metrics proposed in the literature;
- between two annotation assets exploiting the related topics and the well-known metrics on vocabularies or ontologies.

In our model, a similarity hyperedge is effectively generated if $\omega(\vec{e}_i) \geq \gamma$, γ being a given threshold. To better explain this type of relationships, we provide in Figure 2 an example of *multimedia similarity* relationship.



Id = MSN001SMR00001

Name = "Similarity between two sunrise"

Type = Similarity relationship

Timestamp= 21/02/2016 22:31:02

Figure 2. Multimedia similarity relationship

Definition 2.5 (User to Multimedia relationship) *Let $\widehat{U} \subseteq U$ a set of users in a MSN and $\widehat{M} \subseteq M$ a set of multimedia objects, we define* user to multimedia relationship *an hyperedge e_i with the following properties:*

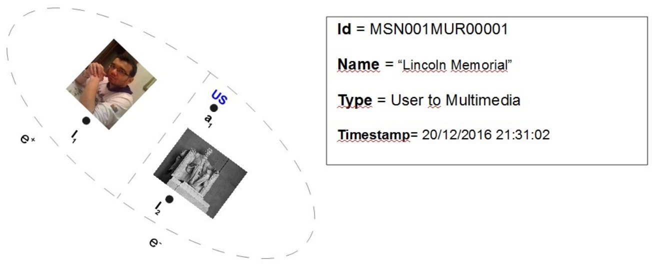
1. $V^+_{e_i} = u_k$ *such that* $u_k \in \widehat{U}$,
2. $V^-_{e_i} \supseteq \widehat{M}$.

The weight function for such relationship reurns as value $\frac{\hat{H}_{kj}}{H_k}$, $\hat{H}_{kj}$ being the average number of distinct user to multimedia social paths between u_k and m_j for each $m_j \in \widehat{M}$, and H_k the number of user to multimedia paths having as initial vertex u_k[2].

Examples of "user to multimedia" relationships are represented, as an example, by *publishing*, *reaction*, *annotation* (in this case the set $V^-_{e_i}$ also contains one or more annotation assets) or *user tagging* (involving also one ore more users) activities. To better explain this type of relationships, we provide in Figure 3 an example of *multimedia tagging* relationship.

[1]In this case ω represents the strengthness of the relationship between two users with respect to the other users.

[2]In this case ω represents the strengthness of the relationship between a user and a given multimedia object with respect to the other objects.



Id = MSN001MUR00001

Name = "Lincoln Memorial"

Type = User to Multimedia

Timestamp= 20/12/2016 21:31:02

Figure 3. Multimedia tagging

2.3 Ranking functions

Ranking functions can be profitably used to "rank" users and multimedia objects in a MSN in an absolute way or with respect to a given topic of interest. Let us first introduce some preliminary definitions.

Definition 2.6 (Distances) *We define* minimum distance *($d_{min}(v_i, v_j)$),* maximum distance *($d_{max}(v_i, v_j)$) and* average distance *($d_{avg}(v_i, v_j)$) between two vertices of a MSN the length of the shortest hyperpath, the length of the longest hyperpath and the average length of the hyperpaths between v_i and v_j, respectively. In a similar manner, we define the* minimum distance *($d_{min}(v_i, v_j\,|v_k)$),* maximum distance *($d_{max}(v_i, v_j\,|v_k)$) and* average distance *($d_{avg}(v_i, v_j\,|v_k)$) between two vertices v_i and v_j, for which there exists a hyperpath containing v_k.*

In the computation of distances, we apply a *penalty* if the considered hyperpaths contain some users: all the distances can be computed as $\tilde{d}(v_i, v_j) = d(v_i, v_j) + \log(\beta \cdot N)$, N being the number of user vertices in the hyperpath between v_i and v_j and β a scaling factor[3].

Definition 2.7 (λ-Nearest Neighbors Set) *Given a vertex $v_i \in V$ of a MSN, we define the* λ-Nearest Neighbors Set *of v_i the subset of vertices NN_i^λ such that $\forall v_j \in NN_i^\lambda$ we have $\tilde{d}_{min}(v_i, v_j) \leq \lambda$ with $v_j \in U$. Considering only the constrained hyperpaths containing a vertex v_k, we denote with NN_{ik}^λ the set of nearest neighbors of v_i such that $\forall v_j \in NN_{ik}^\lambda$ we have $\tilde{d}_{min}(v_i, v_j\,|v_k) \leq \lambda$.*

If we consider as neighbors only vertices belonging to user type, the NN^λ set is called λ-*Nearest Users Set* and denoted as NNU^λ, similarly in case of multimedia objects we define the λ-*Nearest Objects Set* as NNO^λ . On the top of such definitions, we are able to introduce the *ranking functions*.

[3]Such strategy is necessary in the ranking to penalize *lurkers*, i.e.users of a MSN that are quite inactive and not directly interact with multimedia content but through user to user relationships.

Definition 2.8 (User Ranking function) *Given a user $u_i \in U$ and a subset of users $\widehat{U} \subseteq U(u_i \notin \widehat{U})$ of a MSN, a* user ranking function *is a particular function $\rho : U \to [0, 1]$ able to associate a specific* rank *to the user u_i with respect to the community $\widehat{U}$ that is computed as in the following:*

$$\rho_{u_i}\left(\widehat{U}\right) = \frac{\left|NNU_{u_i}^\lambda \cap \widehat{U}\right|}{\left|\widehat{U}\right|} \tag{1}$$

NNU_i^λ being the λ-Nearest Users Set of u_i.

Definition 2.9 (Multimedia Ranking function) *Given a multimedia object $m_i \in M$ and a subset of users $\widehat{U} \subseteq U$ of a MSN, a* multimedia ranking function *is a particular function $\rho : M \to [0, 1]$ able to associate a specific* rank *to the object m_i with respect to the community $\widehat{U}$ that is computed as in the following:*

$$\rho_{m_i}\left(\widehat{U}\right) = \frac{\left|NNU_{m_i}^\lambda \cap \widehat{U}\right|}{\left|\widehat{U}\right|} \tag{2}$$

$NNU_{m_i}^\lambda$ being the λ-Nearest Users Set of m_i.

In a similar manner, considering only hyperpaths containing a given topic a_j we can define the *topic sensitive* user ($\rho_{u_i}^{a_j}\left(\widehat{U}\right)$) and multimedia ($\rho_{m_i}^{a_j}\left(\widehat{U}\right)$) ranking functions.

Definition 2.10 (Topic User Ranking function) *Given a user $u_i \in U$ and a subset of users $\widehat{U} \subseteq U(u_i \notin \widehat{U})$ of a MSN, a* topic user ranking function *is a particular function $\rho_u^a : U \times A \to [0, 1]$ able to associate a specific* rank *to the user u_i with respect to the community $\widehat{U}$ given the topic a_j that is computed as in the following:*

$$\rho_{u_i}^{a_j}\left(\widehat{U}\right) = \frac{\left|NN_{ij}^\lambda \cap \widehat{U}\right|}{\left|\widehat{U}\right|} \tag{3}$$

NN_{uij}^λ being the λ-Nearest Users Set of u_i given a_j.

Definition 2.11 (Topic Multimedia Ranking function) *Given a multimedia object $m_i \in M$ and a subset of users $\widehat{U} \subseteq U$ of a MSN, a* multimedia ranking function *is a particular function $\rho_m^a : M \times A \to [0, 1]$ able to associate a specific* rank *to the object m_i with respect to the community $\widehat{U}$ given the topic a_j that is computed as in the following:*

$$\rho_{m_i}^{a_j}\left(\widehat{U}\right) = \frac{\left|NN_{kj}^\lambda \cap \widehat{U}\right|}{\left|\widehat{U}\right|} \tag{4}$$

NN_{mij}^λ being the λ-Nearest Users Set of m_i given a_j.

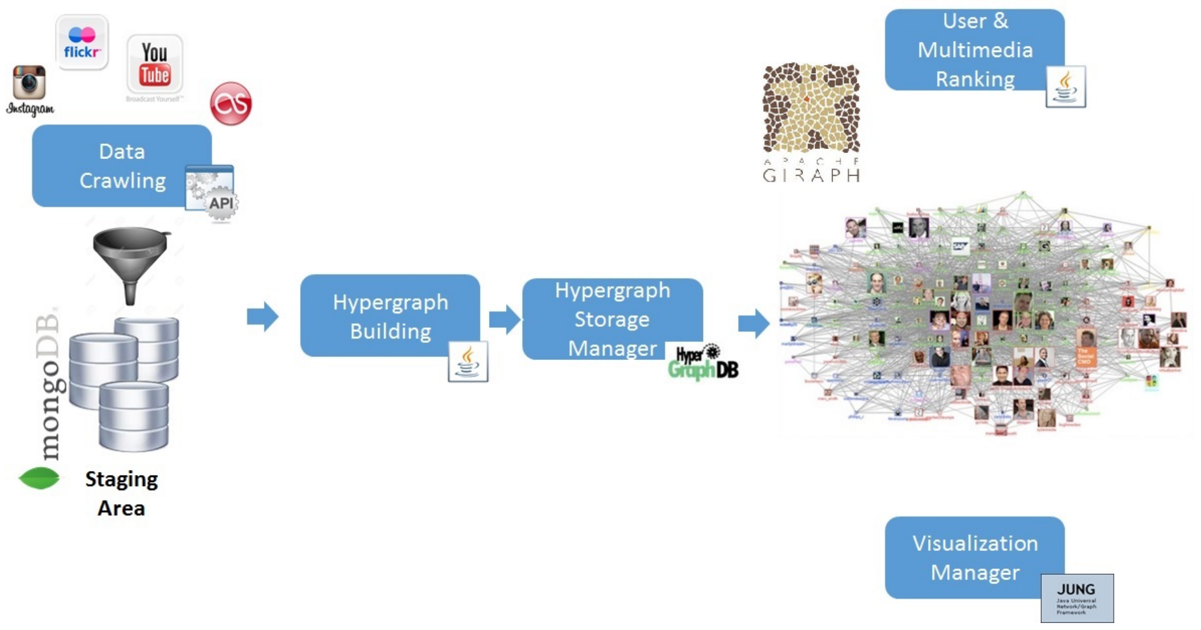


Figure 4. Proposed prototype

In our model the concept of *rank* of a given node is related to the concept of *influence*, and in our vision it can be measured by the number of user nodes that are "reachable" within a certain number of steps using social paths.

By similarity relationships paths can be "implicitly" instantiated: two users (that are not friend, do not belong to any group and do not share any multimedia object) have annotated two images that are very similar, or they have commented two different posts which concern similar topics.

3 Methodology to extract information by Social Network

In order to apply our ranking evaluation we have to extract information about the posts of the customers of an Online Social Network. In this section, we describe our analyses performed on Yelp social network.
The Dataset used for the experimentation is given by Yelp website, it is is composed by: 4.1 millions of reviews, 947 thousands of tips posted by 1 million users for 144 thousands of businesses.

In order to obtain information about each review, we used Gate NLP Tool developed by University of Sheffield (https://gate.ac.uk).

Gate is an open source software able to solve many text-processing problems. This tool is plugin-based so is possible to customize the processing steps adding or removing modules, in order to obtain different results.
Gate components are specialized types of Java Bean and are of three type:

- Language resources (LRs): entities such as lexicons, corpora or ontologies.
- Processing resources (PRs): entities such as parsers, generators or ngram modellers.
- Visual resources (VRs): visualization and editing components

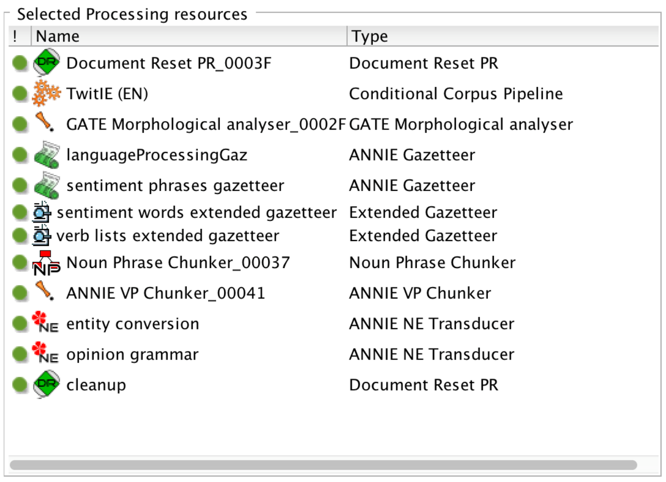
Gate's CORE, for its structure, is named CREOLE: Collection of Reusable Objects for Language Engineering.

Because Yelp reviews are encoded as a set of JSON tuples, a semi-structured data type, we needed to store this dataset into a NoSQL Database. We chose CouchDB, a Document-Oriented Database by Apache Foundation.

CouchDB is a schemaless database with an intuitive HTTP/JSON API. It speaks JSON natively so it is what we needed. To perform analysis on each review, we used an Official plugin of the GATE framework, developed for Twitter.

The pipeline used in this plug-in is composed by:

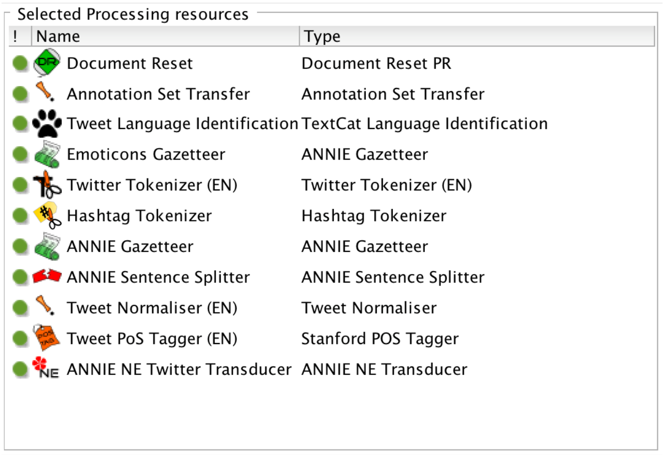
- Document reset:for resetting the default annotation set;
- TwitIE: a pipeline specialized to analyze tweets.
- Gate Morphological Analyzer: taking as input a tokenized GATE document. Considering one token and its part of speech tag, one at a time, it identifies its lemma. LanguageProcessingGaz: an ANNIE Gazetteer. The role of the gazetteer is to identify entity names in the text based on lists.



Selected Processing resources

!	Name	Type
●	Document Reset PR_0003F	Document Reset PR
●	TwitIE (EN)	Conditional Corpus Pipeline
●	GATE Morphological analyser_0002F	GATE Morphological analyser
●	languageProcessingGaz	ANNIE Gazetteer
●	sentiment phrases gazetteer	ANNIE Gazetteer
●	sentiment words extended gazetteer	Extended Gazetteer
●	verb lists extended gazetteer	Extended Gazetteer
●	Noun Phrase Chunker_00037	Noun Phrase Chunker
●	ANNIE VP Chunker_00041	ANNIE VP Chunker
●	entity conversion	ANNIE NE Transducer
●	opinion grammar	ANNIE NE Transducer
●	cleanup	Document Reset PR

Figure 5. Review Analysis Pipeline



Selected Processing resources

!	Name	Type
●	Document Reset	Document Reset PR
●	Annotation Set Transfer	Annotation Set Transfer
●	Tweet Language Identification	TextCat Language Identification
●	Emoticons Gazetteer	ANNIE Gazetteer
●	Twitter Tokenizer (EN)	Twitter Tokenizer (EN)
●	Hashtag Tokenizer	Hashtag Tokenizer
●	ANNIE Gazetteer	ANNIE Gazetteer
●	ANNIE Sentence Splitter	ANNIE Sentence Splitter
●	Tweet Normaliser (EN)	Tweet Normaliser
●	Tweet PoS Tagger (EN)	Stanford POS Tagger
●	ANNIE NE Twitter Transducer	ANNIE NE Transducer

Figure 6. TwitIE Pipeline

- Verb Lists Extended Gazetteer: an extended version of the Gate Default List Gazetteer.

- Noun Phrase Chunker: The NP Chunker application is a Java implementation of the Ramshaw and Marcus BaseNP chunker which attempts to insert brackets marking noun phrases in text which have been marked with POS tags

- ANNIE VP Chunker: The rule-based verb chunker, based on a number of English grammars.

- Entity Conversion, ANNIE NE Transducer: a semantic tagger. It contains rules, which act on annotations assigned in earlier phases, in order to produce outputs of annotated entities.

- Opinion Grammar, ANNIE NE Transducer.

- TwitIE is a specialized pipeline that is composed by many components.

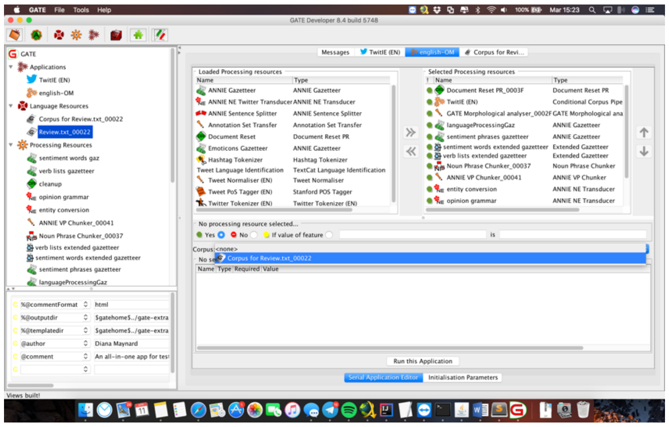
The core of this application is TextCat Language Identification and a huge set of gazetteers customized to recognize hashtags and emojis.
TextCat Language Identification is necessary because our dataset is composed by reviews written in English, French and Deutsch natural language. TwitIE is the lexical and semantic analyzer in our pipeline and its results allow to perform deeper text analysis.

We use the described GATE functionalities in order to analyze the set of reviews. We have to set the documents parameters.

We create a Corpus from the input Documents.

We launch the system functionalities by selecting the Application English-OM and set the corpus to analyze.
After the computation ended, we to check the results Double-click on Document, click on Annotation Sets and Annotation List to view tags.
Each sentence that have a sentiment[18], will be tagged as SentenceSentiment with a set of Features[19], that are customizable using a JAPE Grammar: a set of phases, each of which consists of a set of pattern/action rules. The phases run sequentially and constitute a cascade of finite state transducers over annotations. The left-hand-side (LHS) of the rules consist of an annotation pattern description. The right-hand-side (RHS) consists of annotation manipulation statements.

In order to manage the 4.1 millions of reviews composing the dataset, we created a batch java program that uploaded the reviews as Document on CouchDB. After that, our program perform a HTTP GET request to database to obtain, for each single file, the text of the review, executing Gate on it, load Sentiment parameters and perform the Sentiment Analysis. The last step is to update the Document on CouchDB, performing an HTTP PUT request. The obtained



GATE
Applications
TwitIE (EN)
english-OM
Language Resources
Corpus for Review.txt_00022
Review.txt_00022
Processing Resources
sentiment words gaz
verb lists gazetteer
cleanup
opinion grammar
entity conversion
ANNIE VP Chunker_00041
Noun Phrase Chunker_00037
verb lists extended gazetteer
sentiment words extended gazetteer
sentiment phrases gazetteer
languageProcessingGaz

%@commentFormat	html
%@outputdir	$gatehome$../gate-extra
%@templatedir	$gatehome$../gate-extra
@author	Diana Maynard
@comment	An all-in-one app for test

Loaded Processing resources

Name	Type
ANNIE Gazetteer	ANNIE Gazetteer
ANNIE NE Twitter Transducer	ANNIE NE Transducer
ANNIE Sentence Splitter	ANNIE Sentence Splitter
Annotation Set Transfer	Annotation Set Transfer
Document Reset	Document Reset PR
Emoticons Gazetteer	ANNIE Gazetteer
Hashtag Tokenizer	Hashtag Tokenizer
Tweet Language Identification	TextCat Language Identification
Tweet Normaliser (EN)	Tweet Normaliser
Tweet PoS Tagger (EN)	Stanford POS Tagger
Twitter Tokenizer (EN)	Twitter Tokenizer (EN)

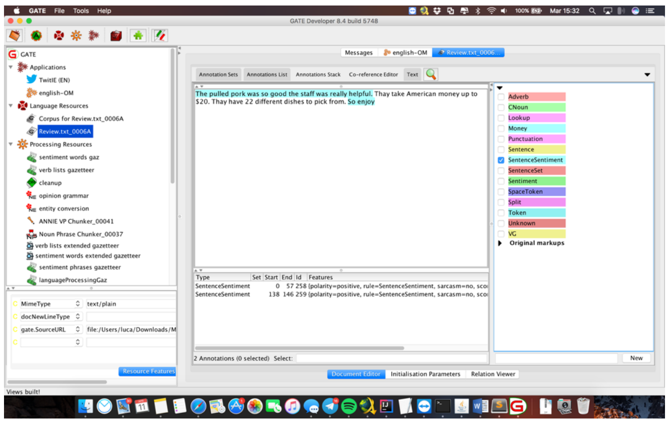
Selected Processing resources

!	Name	Type
●	Document Reset PR_0003F	Document Reset PR
●	TwitIE (EN)	Conditional Corpus Pipe
●	GATE Morphological analyser_0002F	GATE Morphological ana
●	languageProcessingGaz	ANNIE Gazetteer
●	sentiment phrases gazetteer	ANNIE Gazetteer
●	sentiment words extended gazetteer	Extended Gazetteer
●	verb lists extended gazetteer	Extended Gazetteer
●	Noun Phrase Chunker_00037	Noun Phrase Chunker
●	ANNIE VP Chunker_00041	ANNIE VP Chunker
●	entity conversion	ANNIE NE Transducer
●	opinion grammar	ANNIE NE Transducer

No processing resource selected...
[✓ Yes] No If value of feature is
Corpus: <none>
Corpus for Review.txt_00022

Name	Type	Required	Value

Figure 7. Corpus Selection

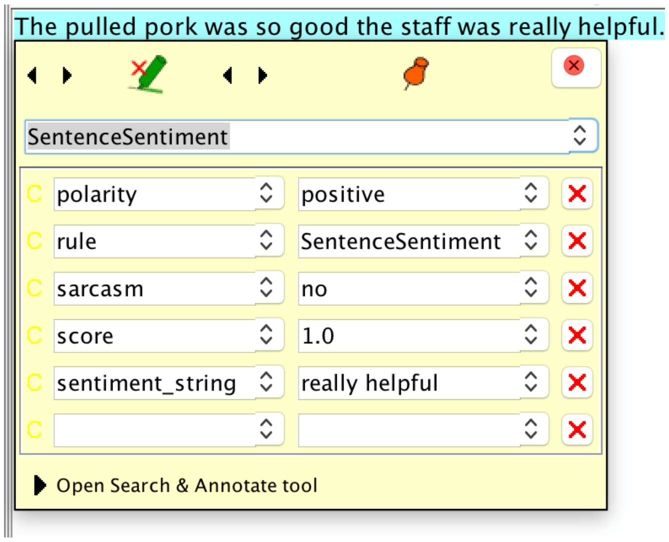


The pulled pork was so good the staff was really helpful. Thay take American money up to $20. Thay have 22 different dishes to pick from. So enjoy

```
Type               Set Start End  Id  Features
SentenceSentiment        0    57  258 {polarity=positive, rule=SentenceSentiment, sarcasm=no, sco
SentenceSentiment      138   146  259 {polarity=positive, rule=SentenceSentiment, sarcasm=no, sco
```

2 Annotations (0 selected) Select:

Figure 8. Document Annotations View



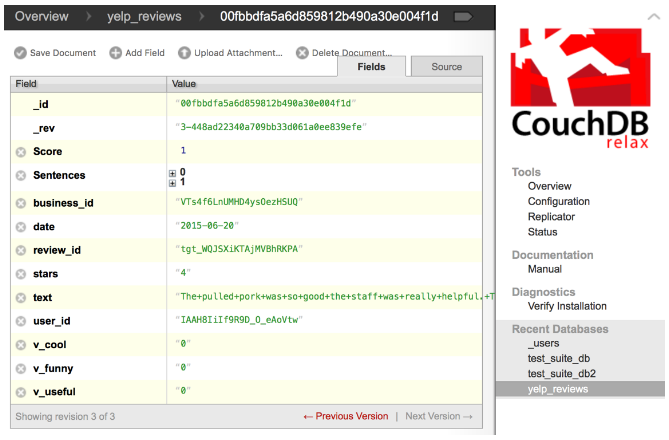
The pulled pork was so good the staff was really helpful.

SentenceSentiment

polarity	positive
rule	SentenceSentiment
sarcasm	no
score	1.0
sentiment_string	really helpful

Open Search & Annotate tool

Figure 9. Information about reviews



Field	Value
_id	"00fbbdfa5a6d859812b490a30e004f1d"
_rev	"3-448ad22340a709bb33d061a0ee839efe"
Score	1
Sentences	0 1
business_id	"VTs4f6LnUMHD4ysOezHSUQ"
date	"2015-06-20"
review_id	"tgt_WQJSXiKTAjMVBhRKPA"
stars	"4"
text	"The+pulled+pork+was+so+good+the+staff+was+really+helpful.+T
user_id	"IAAH8IiIf9R9D_0_eAoVtw"
v_cool	"0"
v_funny	"0"
v_useful	"0"

Showing revision 3 of 3 ← Previous Version | Next Version →

Figure 10. Obtained Output

information are structured in two fields:

Score, representing the sentiment of the entire review. It is the mean value of the single sentences score. Sentences, which is an array of sentences that generated Score.

4 Conclusions and Future Work

In this paper we described a data model for *Multimedia Social Networks*, extracting and modelling information about users. Inspired by hypergraph based approaches, our model provides a solution for representing MSNs sufficiently general with respect to: i) a particular social information network, ii) the different kinds of entities, iii) the different types of relationships, iv) the different applications.

We developed a methodology using a combination of modules applications provided by GATE NLP toolkit, that allows the extraction of relevant information from post related to the online social network Yelp.

As future work, we are planning to exploit the introduced ranking functions to support multimedia recommendation and influence analysis applications, in order to perform an experimental evaluation of the proposed model.

References

[1] F. Moscato, "Exploiting model profiles in requirements verification of cloud systems," *International Journal of High Performance Computing and Networking*, vol. 8, no. 3, pp. 259–274, 2015.

[2] G. Nan, C. Zang, R. Dou, and M. Li, "Pricing and resource allocation for multimedia social network in cloud environments," *Knowledge-Based Systems*, vol. 88, pp. 1 – 11, 2015.

[3] D. Liu, G. Ye, C.-T. Chen, S. Yan, and S.-F. Chang, "Hybrid social media network," in *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012, pp. 659–668.

[4] Z. Zhang and K. Wang, "A trust model for multimedia social networks," *Social Network Analysis and Mining*, vol. 3, no. 4, pp. 969–979, 2013.

[5] X. Ji, Q. Wang, B.-W. Chen, S. Rho, C. J. Kuo, and Q. Dai, "Online distribution and interaction of video data in social multimedia network," *Multimedia Tools and Applications*, pp. 1–14, 2014.

[6] F. T. O'Donovan, C. Fournelle, S. Gaffigan, O. Brdiczka, J. Shen, J. Liu, and K. E. Moore, "Characterizing user behavior and information propagation on a social multimedia network," in *Multimedia and*

Expo Workshops (ICMEW), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–6.

[7] V. Moscato, A. Picariello, and V. Subrahmanian, "Multimedia social networks for cultural heritage applications: the givas project," in *Data Management in Pervasive Systems*. Springer, 2015, pp. 169–182.

[8] B. Di Martino and F. Moscato, "An ontology based methodology for automated algorithms recognition in source code," 2010, pp. 1111–1116.

[9] F. Moscato, "Model driven engineering and verification of composite cloud services in metamorp(h)osy," 2014, pp. 635–640.

[10] F. Amato, A. Mazzeo, V. Moscato, and A. Picariello, "A framework for semantic interoperability over the cloud," in *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*. IEEE, 2013, pp. 1259–1264.

[11] F. Amato, M. Barbareschi, V. Casola, and A. Mazzeo, "An fpga-based smart classifier for decision support systems," in *Intelligent Distributed Computing VII*. Springer, 2014, pp. 289–299.

[12] M. Albanese, A. d'Acierno, V. Moscato, F. Persia, and A. Picariello, "A multimedia recommender system," *ACM Transactions on Internet Technology (TOIT)*, vol. 13, no. 1, p. 3, 2013.

[13] V. Moscato, A. Picariello, and A. M. Rinaldi, "Towards a user based recommendation strategy for digital ecosystems," *Knowledge-Based Systems*, vol. 37, pp. 165–175, 2013.

[14] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "A framework and graphical development environment for robust nlp tools and applications." in *ACL*, 2002, pp. 168–175.

[15] W. Balzano, M. R. Del Sorbo, and S. Stranieri, "A logic framework for c2c network management," in *Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on*. IEEE, 2016, pp. 52–57.

[16] W. Balzano, M. R. Del Sorbo, A. Murano, and S. Stranieri, "A logic-based clustering approach for cooperative traffic control systems," in *International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*. Springer, 2016, pp. 737–746.

[17] W. Balzano, A. Murano, and F. Vitale, "V2v-en–vehicle-2-vehicle elastic network," *Procedia Computer Science*, vol. 98, pp. 497–502, 2016.

[18] F. Colace, M. De Santo, and L. Greco, "A probabilistic approach to tweets' sentiment classification," in *Affective computing and intelligent interaction (ACII), 2013 humaine association conference on*. IEEE, 2013, pp. 37–42.

[19] F. Colace, M. De Santo, L. Greco, V. Moscato, and A. Picariello, "A collaborative user-centered framework for recommending items in online social networks," *Computers in Human Behavior*, vol. 51, pp. 694–704, 2015.

Ergodic Hidden Markov Models for Workload Characterization Problems

Alfredo Cuzzocrea
DIA Dept., University of Trieste and ICAR-CNR, Italy
`alfredo.cuzzocrea@dia.units.it`

Enzo Mumolo
DIA Dept., University of Trieste, Italy
`mumolo@units.it`

Gianni Vercelli
DIBRIS Dept., University of Genova, Italy
`gianni.vercelli@unige.it`

June 3, 2017

Abstract

We present a novel approach for accurate characterization of workloads. Workloads are generally described with statistical models and are based on the analysis of resource requests measurements of a running program. In this paper we propose to consider the sequence of virtual memory references generated from a program during its execution as a temporal series, and to use spectral analysis principles to process the sequence. However, the sequence is time-varying, so we employed processing approaches based on Ergodic Continuous Hidden Markov Models (ECHMMs) which extend conventional stationary spectral analysis approaches to the analysis of time-varying sequences.

In this work, we describe two applications of the proposed approach: the on-line classification of a running process and the generation of synthetic traces of a given workload. The first step was to show that ECHMMs accurately describe virtual memory sequences; to this goal a different ECHMM was trained for each sequence and the related run-time average process classification accuracy, evaluated using trace driven simulations over a wide range of traces of SPEC2000, was about 82%. Then, a single ECHMM was trained using all the sequences obtained from a given running application; again, the classification accuracy has been evaluated using the same traces and it resulted about 76%. As regards the synthetic trace generation, a single ECHMM characterizing a given application has been used as a stochastic generator to produce benchmarks for spanning a large application space.

1 Introduction

Performance evaluation of computer systems requires to test different alternatives under identical conditions. However, a real computing environment is generally not repeatable, and for this reason it is necessary to characterize the workload by developing a workload model that can be used repeatedly. Once a workload model is available, changes in the workload and in the system can be studied under controlled conditions.

As pointed out in [1], workload characterization using a model plays a fundamental role in many areas, namely to understand the key resource usage of applications, to tune computer architectures, to validate trace reduction mechanisms, to guide the selection of programs for obtaining benchmark sets, to generate synthetic traces to span application spaces, and to create abstract program behavior models for perfor-

DOI reference number: 10.18293/DMSVLSS2017-020

mance studies of computer systems.

Workloads are typically modeled as stochastic processes and analyzed with statistical techniques [3] [4]. This is because different benchmarks are obtained from a single application for different inputs, and the only way to describe all the potential application space is through the extraction from the running application of suitable parameters which describes the main features of the workload.

A running application thus produces a huge amount of data; the only way to analyze such data is by means of statistical techniques. In this paper we propose to use ergodic Hidden Markov Models as statistical models of workloads. Our approach is based on the idea to treat the sequences of memory page references produced by a running application as time-varying discrete-time series of data and to analyze them with statistical techniques using spectral parameters. The proposed methodology operates as follows: the page references sequences obtained from a running application is divided into segments of some hundreds of page numbers, and each piece is then described with a vector of spectral parameters. Chunks of references are formed by some hundreds of such vectors; the chunks are then used to estimate the parameters of a Hidden Markov Model. Repeating this operation for each running application, we compute a HMM model of the application. The accuracy of such models has been estimated as quite good.

By considering a number of workloads obtained from the same type of application, and re-estimating the parameters of a single Hidden Markov Model, a statistical model of that type of application can be computed. In this way, we have obtained models for several application types, as described in 1.1. In this paper, models have been used in two ways: to determine to which application type belongs a running application and to generate synthetic traces. Both these points are very important from a computer architecture perspective. As regards the benchmark classification, it is important to note that using our approach the classification is possible in run-time, i.e. during the application execution, since the computational complexity is quite low. As regards the synthetic traces generation, HMMs can indeed be viewed as generators of observations, in our case allowing to cover a large application space for computer architecture studies and designs [5].

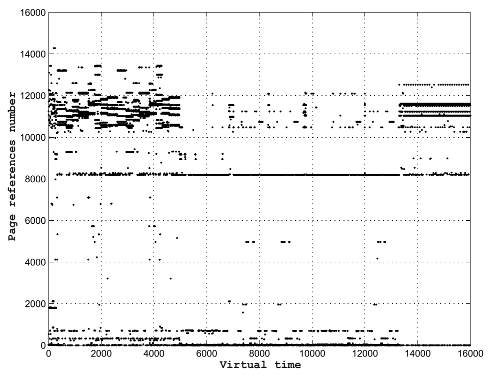


Figure 1: Graphical view of a portion of a sequence of page references.

1.1 Methodology

We used traces-driven simulations to test the proposed approach. The traces were a subset of the SPEC2000 benchmark suite [2], as reported in Tab. 1.

Benchmark	Category	Total number of page references
Bzip2	compression	519960950
Crafty	chess game	322625985
Eon	ray traces	526065045
Gcc	C compiler	646344471
Gzip	compression	477528457
Perl	Perl interpreter	351047065
Twolf	place and route simulator	5246007019
Vpr	FPGA placement and routing	2240811177

Table 1: The traces used in this work.

CPU address traces have been obtained by running the applications of Tab. 1 with different input data; several executions of each application have been considered. The applications of Tab. 1 run on a Pentium 2 processor at 450 MHz under Windows NT operating system. The benchmarks were downloaded from *www.byu.com.* In Fig. 1 a part of a page references trace (16000 virtual time instants) is shown. This figure illustrates the time-varying characteristic of the trace.

The rest of this paper is organized as follows. In Section 2 the properties of HMMs are described together with the considered workload parameters. In Section 3 the workload classification methodologies based on HMM are described while in Section 4 the generation of synthetic traces with HMMs is briefly reported. Finally, in Section 5 some final remarks are reported.

2 Hidden Markov Models for Workload Classification

2.1 Parameters

The page references are produced at a CPU instruction clock rate, because each virtual memory address is translated to a virtual page reference. This information rate is too high to make reasonable workload evaluations, and consequently the number of page references is too large. Therefore, some feature extraction must be performed for getting rid of the redundant information and for reducing the data rate. According to the idea of considering the page references sequence as a signal, we use a spectral description of the page references sequences. Characteristics in the sequences, such as for examples loops or sequential program behaviors, are indeed described in the spectrum. For instance, loops introduce peaks in the spectrum while a sequential address sequence produces a DC component. For example, representing the sequence of Fig. 1 in the log spectral domain, we obtain the data shown in Fig. 2.

Since the page references sequence is time varying, as suggested in Fig. 1, the result of Fig. 2 is obtained with short-time spectral analysis. In particular, the sequence of virtual memory pages is divided into short sections – 120 references long – and analyzed by means of a discrete Fourier transform.

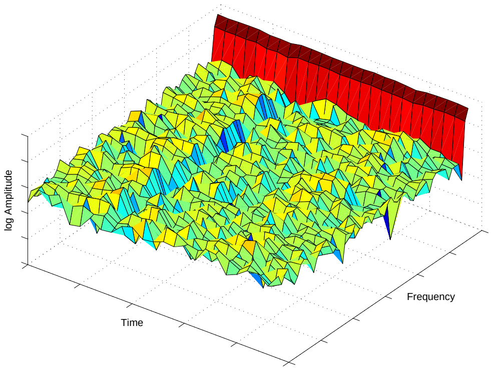


Figure 2: Log-spectral data of the portion of the page references sequence shown in Fig. 1.

It is worth noting that Fig. 2 reports a log-spectral view of the page references trace shown in Fig. 1. In Fig. 2 it is possible to see how the change of behavior in the trace of Fig. 1 at about 5000 virtual time instants reflects in the spectral domain. As in the proposed approach a fundamental issue is related to the comparison between log-spectral data, it is important to define a log-spectral distance between two spectra. To show how to define the log-spectral distance, let us start with the Euclidean distance definition between the log spectra of two sequences, x_n and y_n:

$$\begin{aligned} e(\omega) &= log\,|X(\omega)|^2 - log\,|Y(\omega)|^2 = \\ &= 2(log\,|X(\omega)| - log\,|Y(\omega)|) = \\ &= 2Re\left[log\,(X(\omega)) - log\,(Y(\omega))\right] \end{aligned}$$

where $X(\omega) = \sum_{n=-\infty}^{+\infty} x_n e^{j\omega n}$ is the spectrum of the x_n sequence. On the other hand, $\log(X(\omega)) = \sum_{n=-\infty}^{+\infty} c_n e^{j\omega n}$ where c_n is the cepstrum sequence [7] which is obtained applying an inverse Fourier transform to the log spectrum of the input page references sequence. Hence, calling c_n^x and c_n^y the cepstrum of the x_n and y_n sequences respectively,

$$e(\omega) = 2Re\left[\sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)\, e^{j\omega n}\right] = \sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)\, e^{j\omega n}$$

because the c_n sequences are symmetrical since the input reference page sequence is real. Finally, the spectral distance between two sequences x_n and y_n is

$$d(X,Y) = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^2(\omega)d\omega = \sum_{n=-\infty}^{+\infty} (c_n^x - c_n^y)^2 .$$

In conclusion, the spectral distance between the log spectra is simply the Euclidean distance between the cepstal sequences.

On the basis of this consideration, we described the page references sequences with cepstral coefficients. In Fig. 3 the cepstral representation of the page references sequence of Fig. 1 is reported. As shown in

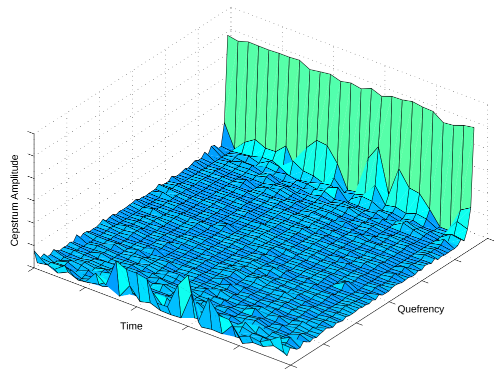


Figure 3: Cepstral description of the portion of the page references sequence shown in Fig. 1.

Fig. 2 the change of trace behavior at about 5000 time instants is reflected in the cepstral domain. In fact, the slow spectral characteristics are seen in the part around zero in the cepstral domain, in Fig. 3 we can see that the initial part of the cepstrum is more spiky around zero reflecting in this way the change of trace behavior seen in Fig. 1. On the basis of that, it is useless to consider all the cepstral coefficient to represent traces; for this reason we used only the first 10 cepstral coefficients.

2.2 Hidden Markov Modeling

Markov models are stochastic interpretations of time series. The basic Markov model is the Markov chain, which is represented with a graph composed by a set of N states; the graph describes the fact that the probability of the next event depends on the previous event. The current state is temporally linked to k states in the past via a set of N^k transition probabilities. Let us denote the generic state of the system with S_t, $S_t \in \{1,2,...,N\}$ and by $a(S_t|S_{t-1}, S_{t-2},\dots,S_{t-k})$ the probability that the system is currently in state S_t given the previously sequence of states $S_{t-1}, S_{t-2},\dots,S_{t-k}$; $a()$ is called the transition probability for a model of order k. In homogeneous Markov chains, the transition probability depend on the previous state only; in such case the transition probabilities can be represented by a transition matrix. If the Markov chain is fully connected, or **ergodic**, each state of the model can be reached from every other state in a single virtual time step. As regards the macroscopic capabilities of such models, we can say that the self loops describe a locality in the process.

Other types of HMMs could better describe the statistical properties of the observed process. For example, the left-to-right models have the property that, as virtual time increases, the state index also increases; they can therefore model sequences whose properties change over time in a successive manner.

In general, an homogeneous Markov chain has the following properties:

1. limited horizon: $Prob(S_{t+1}|S_t, S_{t-1},\dots,S_1) = Prob(S_{t+1}|S_t)$;
2. stationarity: $Prob(S_{t+1}|S_t) = Prob(S_2|S_1)$.

A Markov chain is therefore described by the transition matrix A whose elements are $a_{i,j} = Prob(S_{t+1} = j|S_t = i)$ and the initial probability vector π_i, $\pi_i = Prob(S_1 = i)$, $\sum_{i=1}^{N}\pi_i = 1$. However, in many cases, Markov models are too simple to describe complex real systems and signals [8]. In Hidden Markov Models (HMMs), the output for each state corresponds to an output probability distribution instead of a deterministic event. That is, if the observations are sequences of discrete symbols chosen from a finite alphabet, then for each state there is a corresponding discrete probability distribution which describes the stochastic process to be modeled. In HMMs, the state sequence is hidden and can only be observed through another set of observable stochastic processes. Thus, the state sequence can only be recovered with a suitable algorithm, on the basis of optimization criteria. It is important to note that the observation probabilities has been so far assumed discrete. In many cases, however, the observations are continuous features vectors. It is possible to convert the continuous observations into discrete ones using vector quantization, but in general some performance degradation due to the quantization process is observed. Hence, it is important, from a performance point of view, to use an overall continuous formulation of the algorithms.

Generally speaking, HMMs lead to the three basic problems:

1. the estimation problem: given the observed sequence $\mathbf{O}=O_1, O_2, \ldots, O_T$, how the model parameters λ can be adjusted to maximize $Prob(\mathbf{O}|\lambda)$? This problem concerns the estimation of the model parameters. This estimation process is performed by iteratively maximize the likelihood $Prob(\mathbf{O}|\lambda)$ using an Expectation Maximization (EM) approach [9]. The differences between discrete and continuous HMMs lead to different re-estimation algorithms for the model parameters.

2. the evaluation problem: given the observed sequence $\mathbf{O}$, the problem is to compute the probability that the observed sequence whose produced by the model. This problem can be also stated as follows: given several HMMs and a sequence of observations, how do we choose the model which best matches the observations?

3. the decoding problem: given the observation sequence $\mathbf{O}$, what is the most likely state sequence $S = S_1, S_2, \ldots, S_T$? The decoding is usually performed using the Viterbi algorithm.

3 Workload Classification

For dynamic characterization of processes, the address field of the BYU traces has been extracted. In this way we have obtained a sequence of virtual addresses generated by the processor during the execution of the processes. For converting the trace of addresses into trace of virtual pages, the sequence of addresses has been divided by the page dimension, which we set to 4096 bytes.

Once the sequence of virtual pages has been obtained from every BYU trace and thus for every process, we have tried to use discrete HMMs for their classification. Even if the sequence of pages is a discrete sequence, it can not be used for processes classification using discrete HMMs, as it contains a too high number of symbols.

In order to face this problem, the sequence of virtual pages has been turned into a sequence of few symbols, without loosing meaningful data. The sequence of virtual pages has been turned into sequence of cepstral coefficients by the short time analysis process described in Sec. 2.1.

3.1 Single Trace Classification

The sequences of cepstral coefficients are real number sequences. For analyzing cepstral sequences using a discrete HMM, vector quantization is needed. In this process some degradation is introduced and the training lacks its efficiency.

A continuous HMM can use an input sequences of 10-dimensional cepstral vectors and vector quantization is not needed. The results obtained in this way usually perform better than using the discrete model.

The multivariate Gaussian density is used for describing the cepstral observation. The 10-dimensional cepstral vector is described using a multivariate density having 10 dimensions, and it is specified by means of the mean and covariance matrixes. Using this approach it is supposed that the 10-cepstral coefficients are uncorrelated and so the covariance matrix is diagonal.

In order to choose the number of states and the topology of the HMMs, several tests have been performed. The number of states needed is lower than in the discrete HMM. Considering topology, ergodic models score better results.

In Fig. 4 a graphical representation of the mean classification of all the traces over the number of states for ergodic and left-right models is depicted.

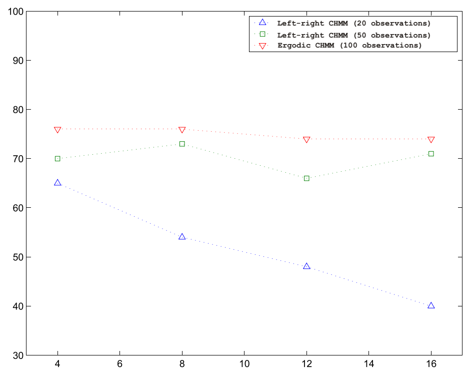


Figure 4: Average recognition rate for all the traces over the number of states of ergodic and left-right models.

As the models using 4 states provides better results using a lower number of observation, we have repeated experiments using this configuration increasing the number of observations.

Using 100 observations for every model, in the ergodic case the recognition mean of single traces is about 82%, in the left-right case this mean is 65%. In Fig. 5 and in Fig. 6 these results are depicted, gathering the traces per workload and computing for every traces group the mean recognition rate.

The ergodic continuous HMMs have been trained using 100 observations for every model. The recognition rate varying the number of states and using all the traces is reported in Fig. 6.

The results obtained using such statistical models demonstrated the effectiveness of this dynamic processes modeling approach. Cepstral coefficient obtained from the virtual pages sequences are a good parameter for describing traces of programs during execution.

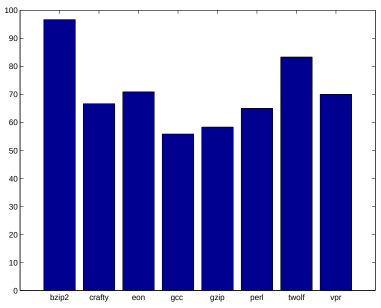


Figure 5: Average classification rate for all the traces using 16-state ergodic discrete HMMs.

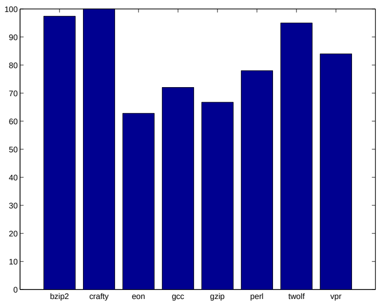


Figure 6: Average classification rate for all the traces with 4-state ergodic continuous HMMs.

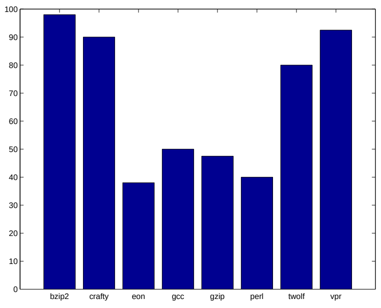


Figure 7: Average classification rate for all the traces with 4-state left-right continuous HMMs.

3.2 Program Behavior Modeling

Dynamic classification of BYU traces, taking as parameter the virtual pages, has obtained satisfactory results. As seen in 3.1, the traces of a single application have been obtained processing such application with different inputs, or processing different functions of the same program.

Then, we have classified the workloads, gathering the traces of the same workload using a single HMM trained with several traces representing the same workload.

Using several traces of the same workload for classifying program behavior using ergodic discrete and continuous HMM, we have obtained the results reported in Fig. 8 and in Fig. 9.

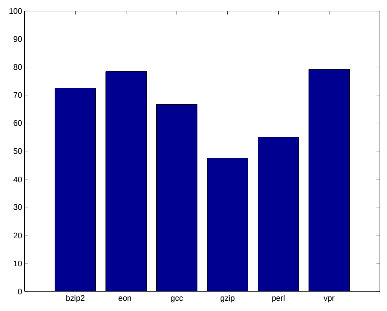


Figure 8: Workload classification using ergodic discrete HMM.

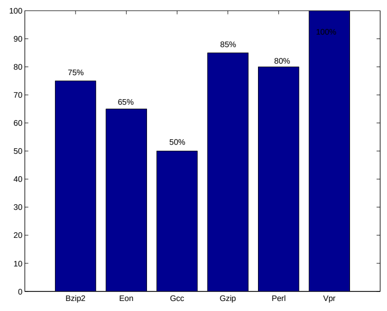


Figure 9: Workload classification using ergodic continuous HMM.

The mean results obtained in the case of ergodic discrete and ergodic continuous HMMs are reported in Tab. 2: ergodic continuous models obtain better classification accuracy than the discrete ones.

	Ergodic Discrete HMM	Ergodic Continuous HMM
Cepstral	65%	76%

Table 2: Workloads classification.

4 Synthetic Trace Generation

A Hidden Markov Model can be used as a generator of a stochastic process. The procedure is the following:

1. Choose an initial state i according to the initial distribution π.
2. Set $t = 1$.
3. Generate a N-dimensional random variable according to the characteristic of the multivariate Gaussian distribution in state i.
4. Perform a state transition according to the transition probabilities $a_{i,j}$.
5. Set $t = t + 1$. If $t < T$ go to 3, else terminate.

The random variable generated in step 3 is a vector of cepstral coefficients. This vector must be inverted to obtain a set of page references.

A result is reported in Fig. 10, where the log-spectral data of a synthetic trace produced with the above procedure and the HMM trained with the trace of Fig. 1 is reported. Fig. 10 should be compared with Fig. 2.

5 Conclusions and Future Work

In this paper we describe an approach for workload characterization using ergodic hidden Markov mod-

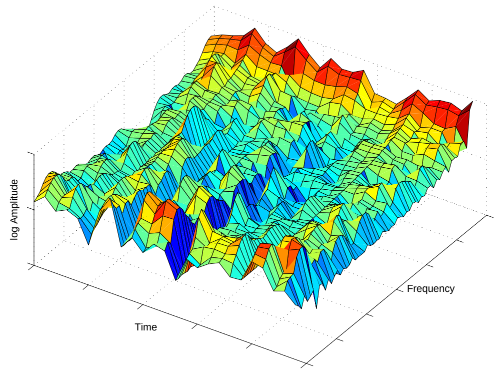


Figure 10: Example of synthetic trace generated using a continuous ergodic HMM represented in the spectral domain.

els. The page references sequences produced by a running application are divided into short virtual time segments and used to train a HMM which models the sequence and is then used for run-time classification of the application type and for synthetic traces generation. The main contribution of our approach are on one hand that a run-time classification of the running application type can be performed and on the other hand that the applications behavior are modeled in such a way that synthetic benchmarks can be generated. Using trace-driven simulation with SPEC2000 benchmarks, the mean classification rate is about 82% for each traces and about 76% using a single HMM to model a single application type. Many future developments of our approach are possible since what we propose in this paper – to use time-varying non-linear processing techniques to treat sequences produced by programs during execution – is a novel approach in computer architecture studies. In addition to this, we believe that another interesting line of research is represented by the adaption of the proposed framework to novel *big data trends* (e.g., [10, 11, 12]).

References

[1] L.K. John, P. Vasudevan, J. Sabarinathan, *Workload characterization: motivation, goals and methodology.* Workload Characterization: Methodology and Case Studies, 1998 29 Nov. 1998 Page(s):3 - 14

[2] C. Niki, J. Thornock, K. Flanagan, *Using the BACH Trace Collection Mechanism to Characterize the SPEC2000 Integer Benchmarks.* Proceedings of the Third IEEE Annual Workshop on Workload Characterization, 2000.

[3] M. Calzarossa, G. Serazzi, *Workload Characterization: A Survey.* Proceedings of the IEEE, vol. 81(8), 1993.

[4] K.J. McDonell, *Benchmark Frameworks and Tools for Modelling the Workload Profile.* Performance evaluation 22, 1995.

[5] J.P. Singh, H.S. Stone, D.F. Thiebaut, *A Model of Workloads and Its Use in Miss-Rate Prediction for Fully Associative Caches.* IEEE Transactions on Computer, vol.41(7), 1992.

[6] L.R. Rabiner, *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* Proceedings of the IEEE, vol.77(2), 1989.

[7] L.R. Rabiner, B.H. Juang, *Foundamentals of Speech Recognition.* Prentice Hall Signal Processing Series, 1993.

[8] Y. Bengio, *Markovian Models for Sequential Data.* Neural Computing Surveys 2, 1999.

[9] J.A. Bilmes, *A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models.* Technical Report, TR-97-021, 1997.

[10] A. Cuzzocrea, *Privacy and Security of Big Data: Current Challenges and Future Research Perspectives.* ACM PSBD 2014 Proceedings, 2014.

[11] A. Cuzzocrea, U. Matrangolo, *Analytical Synopses for Approximate Query Answering in OLAP Environments.* DEXA 2004 Proceedings, 2004.

[12] A. Cuzzocrea, G. Fortino, O.F. Rana, *Managing Data and Processes in Cloud-Enabled Large-Scale Sensor Networks: State-of-the-Art and Future Research Directions.* CCGRID 2013 Proceedings, 2013.

Work in Progress: Identifying and Analyzing Original Projects in an Open-Ended Blocks Programming Environment

Franklyn Turbak, Eni Mustafaraj, and Maja Svanberg
Computer Science Department, Wellesley College
Wellesley, Massachusetts, USA
Email: {fturbak,emustafa,msvanber}@wellesley.edu

Michael Dawson
Independent Researcher
Waltham MA, 02451, USA
Email: mijoda@gmail.com

Abstract—**Tens of millions of people have used online blocks programming environments like App Inventor to learn how to program and build personally meaningful programs and apps. We want to improve blocks programming environments and pedagogies by using learning analytics to identify common problems and then address them. For most users, there is no information about which projects are *original* (built from scratch by individuals or groups based on their own ideas and current programming skills) vs. *unoriginal* (based on tutorials, class exercises, etc.). To understand what App Inventor users are learning and what misconceptions they have, we need to filter out unoriginal projects and focus on original ones.**

Here we describe two key aspects of our work in progress towards this goal. First, we have developed feature-vector representations of App Inventor projects that formalize a notion of structural similarity between them. This representation facilitates filtering out unoriginal work like tutorials and can be used within a group of learners to distinguish classroom activities from original projects. Second, we have developed a graph clustering technique based on project creation timestamps to discover groups of App Inventor users that appear to be taking a course together — essential information for distinguishing original vs. unoriginal work that is not explicitly represented in our datasets.

I. Introduction

In blocks programming environments, programs are assembled out of fragments shaped like jigsaw puzzle pieces. Because they lower barriers to programming [1], these environments, which include App Inventor, Scratch, Snap!, Blockly, Pencil Code, and Alice, have become popular ways for beginners to learn programming concepts and for casual programmers like scientists and hobbyists to write programs.

For example, MIT App Inventor democratizes mobile app creation by empowering those without previous programming or app-building experience to build their own apps [2]. In App Inventor, an Android mobile app can be created in two stages in an online browser-based visual programming environment. First, the user interface components (e.g., buttons, labels, text boxes, images, canvases) and functional components (e.g., camera, sound recorder, GPS location sensor, speech-to-text converter, speech recognizer) of the app can be configured using a drag-and-drop editor. Second, the behavior of the app is specified by connecting visual blocks that correspond to abstract syntax tree nodes in a traditional programming language. Some blocks represent events, conditions, or actions for a particular app component (e.g., the button has been pressed, take a picture with the camera) while others represent standard programming concepts (variable getters and setters, conditionals, loops, procedures, lists, etc.) Nearly 5 million registered App Inventor users have created over 19 million apps, and there are over 350 thousand active monthly users.

Since 2009, the first two authors have used App Inventor in numerous introductory courses, faculty workshops, and other activities. In our experience, App Inventor does lower barriers to making mobile apps. But users often have trouble making their apps work as desired. Some problems are rooted in computational thinking bugs. For example, the state variables of a loop are often improperly initialized in a way that allows it to behave correctly the first time it is run but not on subsequent runs. Working aspects of an app can be implemented in overly complex and roundabout ways. App Inventor programmers often make multiple copies of existing blocks and screens in situations where abstraction mechanisms like procedures and screen templates filled by data would avoid such duplication.

Our long-term goal is to use learning analytics to identify difficulties encountered by blocks programmers and to alleviate these difficulties by improving the programming environments and their associated pedagogies. Towards this goal, we are currently analyzing two datasets of App Inventor users collected in the 27 months between Dec. 2013 and Feb. 2016: all projects of 10 thousand randomly chosen users, and all projects of the 46,320 so-called prolific users, who have created 20 or more projects.

The open-ended nature of App Inventor and lack of information about its users presents many challenges for our research. App Inventor collects no demographic data on users other than what is provided in an optional survey completed by only a small percentage of users. For most users, we have no information on their gender, age, geographic location, programming background, etc. App Inventor accounts and projects normally have an email address, but these have been

DOI reference number: 10.18293/DMSVLSS2017-021

removed from our two large datasets as part of deidentifying them. Importantly, there is no information associated with users or projects that explicitly indicates whether a user took an App Inventor course or whether a project was created as part of a course or other coordinated activity.

In order to understand conceptual difficulties with App Inventor, we want to distinguish *original projects*, in which users create a project from scratch or significantly enhance an existing project based on their own ideas and current programming skills, from *unoriginal projects*, in which users create a project by following the steps of an online tutorial or a guided classroom exercise. We make this distinction for two reasons: (1) filtering out the unoriginal projects of users lets us focus on their skill progression and the misconceptions they have when building original projects; and (2) we can see whether constrained activities like tutorials and classroom exercises involving a particular concept help users with that concept in subsequent open-ended activities.

II. Feature Vectors for App Inventor Projects

When applying a learning analytics lens to how users learn and use App Inventor, it is helpful to formalize a notion of structural similarity between their mobile app projects. This notion facilitates filtering out unoriginal work like tutorials when analyzing projects for computational thinking and promises to be more effective than attempts (e.g., in [3]) based on project names.

One way to determine structural similarity between two App Inventor programs is to focus on their abstract syntax trees, and measure (1) which nodes appear in both trees and (2) which parent-child relationships appear in both trees. This is the basis of the *particle analysis* method developed by Sherman for determining how far away a student's project is from a known desired solution [4]. However, comparing parent-child relationships between two trees is expensive, leading to structural comparisons that are likely to be too slow for analyzing datasets involving millions of programs.

Instead, we represent App Inventor projects as feature vectors, where features include the types of components and blocks used in the program. (App Inventor has dozens of components and over a thousand types of blocks, though a typical program uses only a small subset of these.) Using feature vectors has the advantage that we can leverage standard Python data analysis libraries to determine similarity between projects by calculating distances between vectors. We are still experimenting with various dimensions of this feature vector representation to find the one that best suits our needs. For example: should the features include both component and block types or just block types (since the blocks themselves sometimes contain component information)? Do we simply care whether a feature is present or not, or do we want frequency counts for each feature? Should we give less weight to more common features (known as *term frequency–inverse document frequency (TF-IDF)* in the information retrieval literature)? What is the best way to measure distances between feature vectors in n-dimensional space? So far our experiments indicate that the generalized Jaccard metric (which divides the intersection of feature frequencies by their union) is better than the Euclidean and Manhattan distance metrics.

We have used the feature vector representation of App Inventor projects as the basis for hierarchically clustering the 902 projects created by 16 students in an App Inventor CS0 course at our institution [5]. These clusters provide a way to automatically distinguish original from unoriginal projects in a class setting. When projects done by many students are clustered closely together, we consider these projects to be unoriginal classroom activities. Projects dissimilar to other projects are considered original, as are projects of a single student or pairs of students that are clustered closely together (because they are likely to be different versions of original individual or pair projects). The automatic original vs. unoriginal categorization of projects by our algorithm closely matched the manual labelings we had given them.

III. Discovering Courses of App Inventor Users

The hierarchical clustering technique described above for distinguishing unoriginal classroom activities from original projects requires knowing which users were taking a course together. But in our App Inventor user datasets, there is no explicit information about which users might be associated with a course. Nevertheless, projects do carry a timestamp indicating when they were created. We have developed a graph clustering technique for co-occurring temporal events that leverages these timestamps to discover groups of users who appear to be taking the same course.

The key assumption underlying our technique is that students in a course are physically co-located in a classroom, receiving instruction from an instructor, who often guides students in creating a project within a relatively short time interval. If two users create a project around the same time, this is evidence that they might be in the same course, but it is not conclusive since this can happen by chance. But because courses are typically taught over many weeks or months, there are many opportunities for classmates to create programs around the same time, and two users who share many project creation times are likely to be taking a course together.

In one test of our technique, we selected the subset of the prolific users who created their first project between Aug 15 and Sep 15, 2015. This Fall15 group contained 6012 users. To find co-occurring events, pairs of project timestamps and users were created and stored into a single list that was sorted by the temporal information. Then time windows of plus/minus five minutes were pivoted on every list item to find the co-occurring projects in the interval. This information was used to create graphs in which the nodes are users, edges indicate that two users created at least one co-occurrent project, and edge weights represent the frequency of co-occurrences. After experimenting with raw frequencies, we decided to use proportional frequencies that take into account the total number of projects created by a user. The Fall15 graph was large (2,005,796 edges) even though we left out users who were using the system but were not part of the

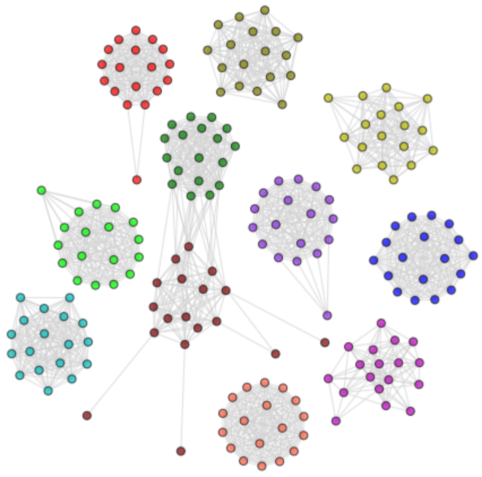


Fig. 1. Eleven clusters, all with 18 members, discovered for the Fall15 data. The cluster colored in red corresponds to the user accounts for all students in a course taught by the first author.

Fall15 group. After several experiments, we decided to filter out edges with five or less co-occurrences, given that students in our courses had 15 to 25 co-occurrences in a semester.

After the filtering process, we use the MCL graph clustering algorithm [6] on the Fall15 graph. This found 462 clusters, which varied in size from 1 to 84 nodes. Figure 1 depicts eleven of these clusters, all of size 18. We show these particular clusters, because one of them belongs to a course taught by the first author. This cluster (in red, top most-left) correctly contains all 16 registered students in the class, the one auditor sitting in on the class, and a previously unknown-to-us private account used by one of the students (the lonely node trailing the cluster).

IV. Current Status and Future Work

We are fine-tuning the choices for project feature vectors and similarity metric in the context of identifying projects in our datasets that appear to be created by following online tutorials. We plan to manually label several hundred projects by their tutorial status, and determine choices that maximize the correct identification. We will then identify tutorials in both our random and prolific datasets and compare some basic statistics between them. E.g., does one group have a higher percentage of tutorial projects than another? Are some tutorials more popular in one dataset than another?

We are also investigating extending our Fall15 course discovery experiment to our entire prolific dataset, and developing ways to verify that the clusters correspond to users taking a course together. A preliminary analysis shows that co-occurring projects within a cluster tend to occur at regular weekly times, bolstering the conclusion that they were created in a course held at these times. We also plan to measure the similarity of the co-occurring projects.

Once we have discovered courses for our prolific dataset, we will use hierarchical clustering to categorize projects that appear to be classroom activities. After filtering out tutorials and classroom exercises, we will be left with original projects that will form the basis of our learning analytics work. One form of unoriginality we do not know how to handle is determining whether any user projects are based on projects from the App Inventor Gallery, a collection of projects shared by members of the user community.

In our analysis of original projects, we plan to build upon the work of Xie and Abelson [7] to study the skill progression of users in their App Inventor projects. We also plan to investigate misconceptions and poor programming style, and see whether these improve over time or are affected by previously completed tutorials and classroom activities. In particular, we will study the usage of App Inventor's abstraction mechanisms (procedures, lists, loops, and generic components), which preliminary observations indicate are used surprisingly infrequently, even among prolific users.

As part of our work, we will develop algorithms to detect common bugs and lack of abstraction in App Inventor programs. We later plan to integrate such detection algorithms within the App Inventor environment itself to provide feedback directly to users about ways to improve their programs. Such detection algorithms could also be used within a teacher dashboard for App Inventor (such as the one sketched in [4]) to highlight students who need help with their code.

We have seen that prolific users may have several similar original projects that appear to be different versions of the same project. We suspect these projects are manual backups of a single project made by users who fear losing their work. We can use hierarchical clustering on a user's original projects to find such versions and see how common this "manual versioning" is in practice. Currently, no history is recorded for App Inventor projects indicating how they evolve over time (though Sherman has developed a fine-grained recording mechanism that may be integrated into App Inventor in the future [4]). So a sequence of versions could provide a useful coarse-grained window on the history of certain projects.

Acknowledgments

This work was supported by Wellesley College Faculty Grants and by the National Science Foundation under grant DUE-1226216.

References

[1] D. Bau, J. Gray, C. Kelleher, J. S. Sheldon, and F. Turbak, "Learnable programming: Blocks and beyond," *Communications of the ACM*, 2017, to appear.

[2] D. Wolber, H. Abelson, and M. Friedman, "Democratizing computing with App Inventor," *GetMobile: Mobile Computing and Communications*, vol. 18, no. 4, pp. 53–58, Jan. 2015.

[3] B. Xie, I. Shabir, and H. Abelson, "Measuring the usability and capability of App Inventor to create mobile applications," in *3rd International Workshop on Programming for Mobile and Touch*, 2015, pp. 1–8.

[4] M. Sherman, "Detecting student progress during program activities by analyzing edit operations on their blocks-based programs," Ph.D. dissertation, University of Massachusetts Lowell, Apr. 2017.

[5] E. Mustafaraj, F. Turbak, and M. Svanberg, "Identifying original projects in App Inventor," in *Proceedings of the 30th International FLAIRS Conference*, May 2017.

[6] S. van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, May 2000.

[7] B. Xie and H. Abelson, "Skill progression in MIT App Inventor," in *IEEE Symposium on Visual Languages and Human-Centric Computing*, 2016, pp. 213–217.

Authors' Index

Program Committee Reviewers' Index

Flora Amato	University of Naples
Danilo Avola	Sapienza University of Rome
Arvind Bansal	Kent State University
Paolo Bottoni	Sapienza University of Rome
Loredana Caruccio	University of Salerno
Shayok Chakraborty	Arizona State University
Maiga Chang	Athabasca University
Shikuo Chang	Uinversity of Pittsburgh
Peter Chapman	Edinburgh Napier University
Mauro Coccoli	DIBRIS - University of Genoa, Italy
Francesco Colace	Università degli studi di Salerno
Kendra Cooper	Independent
Gennaro Costagliola	Dipartimento di Informatica, Università di Salerno
Gennaro Costagliola	University of Salerno
Alfredo Cuzzocrea	ICAR-CNR and University of Calabria
Sergiu Dascalu	University of Nevada, Reno
Andrea De Lucia	Department of Management & Information Technology - University of Salerno
Aidan Delaney	University of Brighton
Vincenzo Deufemia	Department of Computer Science, University of Salerno
Tiansi Dong	Bonn-Aachen International Center for Information Technology B-IT
Henry Duh	Latrobe University
Nathan Eloe	Northwest Missouri State University
Martin Erwig	Oregon State University
Filomena Ferrucci	Università di Salerno
Andrew Fish	University of Brighton
Vittorio Fuccella	Dipartimento di Matematica e Informatica - Università di Salerno
Kaori Fujinami	Tokyo University of Agriculture and Technology
David Fuschi	Bridging Consulting Ltd
Ombretta Gaggi	Dept of Mathematics, University of Padua
Angelo Gargantini	University of Bergamo
Angela Guercio	Kent State University at Stark
Carlos A. Iglesias	Universidad PolitÃcnica de Madrid
Pedro Isaias	Universidade Aberta
Kamen Kanev	Shizuoka University
Jun Kong	North Dakota State University
Yau-Hwang Kuo	National Cheng Kung University
Robert Laurini	INSA Lyon
Jennifer Leopold	Missouri University of Science & Technology
Fuhua Lin	Athabasca University
Hong Lin	University of Houston-Downtown
Alan Liu	National Chung Cheng University
Paolo Maresca	Università di Napoli Federico II
Luana Micallef	Helsinki Institute for Information Technology HIIT

Mark Minas	Universität der Bundeswehr München
Paolo Nesi	University of Florence, DISIT Lab
Max North	Southern Polytechnic State University
Joseph Pfeiffer	New Mexico State University
Antonio Piccinno	University of Bari
Yong Qin	Beijing JiaoTung University
Elvinia Riccobene	Computer Science Dept., University of Milan
Michele Risi	University of Salerno
Peter Rodgers	University of Kent
Veronica Rossano	Depatment of Computer Science - University of Bari
Chaman Sabharwal	Missouri University of Science and Technology
Giuseppe Santucci	University of Rome
Gem Stapleton	University of Brighton
Franklyn Turbak	Wellesley College
Giuliana Vitiello	Dipartimento di Matematica e Informatica- Università di Salerno
Atsuo Yoshitaka	Japan Advanced Institute of Science and Technology
Tomas Zeman	Czech Technical University in Prague
Kang Zhang	University of Dallas, Texas
Zhigao Zheng	Central China Normal University

External Reviewers' Index

Case, Denise M.
Eloe, Nathan
Fu, Yanjie