A Performance Comparison of Graph Analytic Methods for Supporting Enterprise Architecture Model Maintenance

Nikhitha Rajashekar[1], Simon Hacks[2], and Nuno Silva[3]

[1] Research Group Software Construction, RWTH Aachen University, Germany
nikhitha93@gmail.com hacks@swc.rwth-aachen.de
[2] Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden
shacks@kth.se
[3] Department of Computer Science and Engineering, Technical University of Lisbon, Lisbon, Portugal
nuno.silva@tecnico.ulisboa.pt

Abstract. Enterprise Architecture (EA) models help enterprise architects to make business decisions and to support organisations to understand and analyse its structure. Creating and maintaining such an EA model is expensive and time-consuming in large organisations. In this paper, the authors provide a review of how graph analytic methods can be used to support EA model maintenance. To achieve the best results for the duplicate identification, it is necessary to evaluate and optimise different graph analytic and Machine Learning (ML) algorithms to support the maintenance process.

Keywords: Enterprise Architecture · Model · Maintenance · Graph · Machine Learning

1 Introduction

The practice of Enterprise Architecture (EA) is, inter alia, about providing a blueprint and roadmap for aligning business strategy with IT [14]. Usually, in large organisations, it is an effortful task for a single individual to understand how the various enterprise components connect and work together. Thus, EA can be used to create a visual representation of the enterprise system.

In todays organisations, it is necessary to deliver insights quickly and analyse large amounts of enterprise data. In this context, it is essential to understand and model the EA components and relationships. EA models are comprised of a set of components, representing the enterprise's assets, and the links between them. This interpretation allows to understand EA models as graphs. Graph-based tools provide an advantage to enterprise architects by providing a unified view of EA that helps in identifying and understanding it [27, 21].

To provide an effective analysis of the enterprise, the relying EA repository needs to be sound and up-to-date. Maintaining an EA model that is up-to-date and consistent with the enterprise can be an effortful task due to the size

CEUR-WS.org/Vol-2586/paper5.pdf

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

and complexity of EA models, frequent changes in the architecture, and the challenges of collecting and managing EA information from different stakeholders [10]. Due to the increase in information systems, as a manifestation of the digital transformation age, the documentation of enterprise information as an inherent part of EA model design is often regarded as time-consuming, cost-intensive, and error-prone [9, 8, 15].

Previously, we have elaborated on different Machine Learning (ML) techniques to support enterprise architects by identifying possible duplicates within EA models, which cause an unnecessary expansion [4]. However, in our previous research, we identified potential algorithms, which can support the enterprise architects, but we did not focus on the run time of the algorithms. Those long computation times prevent our approach to be adapted in working environments. To cope with this challenge, we set up a ML study addressing the following research question:

Which graph analytic method performs best to identify duplicate components in EA models?

In line with the above, we compare different approaches which address a specific kind of challenge in EA model maintenance: the identification of duplicate components in the EA repository. In the following sections, we first discuss the related literature. We then present the case study including the conducted research methodology and introduce the different graph analytic approaches to investigate the EA models. Finally, we illustrate the experimental results, point out the identified limitations, and conclude our work.

2 Related Work

Previous research [7, 27, 29] was motivated by analysing EA models as a graph and applying different ML concepts on EA models by providing decision support for enterprise architects. The work of Dreyfus and Iyer [7] focuses on representing the complexity of information systems architecture in social network terms and then capturing insights from the graph representation.

Different similarity approaches have been proposed in the literature to identify the similarity and the differences between models and their components to be matched. The work of Dijkman et al. [6] presents three similarity metrics to investigate matching of similar business process models in a given repository, namely (i) structural similarity that compares element labels by considering topology structure of business process models; (ii) behavioural similarity that compares element labels by considering behavioural semantics of process models; and (iii) label matching similarity that compares elements based on words in the labels of business process model elements (string edit distance).

The work of Aier and Schnherr [2] presents a clustering approach in determining the structure of Service Oriented Architectures (SOA). The paper shows the application of clustering algorithms in supporting the design of a SOA. However, the publication does not present the evaluation criteria for comparing different clustering methods. Discovering communities or clusters have been extensively

studied in various research areas [11, 25]. Several research papers discuss the comparison of community detection algorithms [22, 23, 32].

Our research is also linked to our to the field of model reuse for UML diagrams. Ganser et al. [12] developed an approach to encourage the usage of models or rather parts of models. Therefore, they transform the diagrams into graphs and store them in a library to recommend them to the modeller if necessary.

3 Research Approach

As already stated, we like to identify the best performing algorithm to identify duplicate components in EA models. Therefore, we compare different ML algorithms inspired by the ML life-cycle model provided by [28]. The main idea is to improve the performance of a model by comparing different algorithms and metrics:

1. **Raw data:** Any EA model can be expressed in XML, which is mainly used for storage and communication purposes. Thus, we first acquire our EA model in XML format, which serves as input for our approach.
2. **Data preprocessing:** This step includes transforming raw data into a relevant format and cleaning the data set. We parse the raw data obtained from elements and relations files to a graph data object using igraph library [5]. Next, we normalise the data to standardise the range of independent variables and its features.
3. **Feature engineering:** The next step is to extract or compute relevant graph feature statistics. The features extracted from data will directly influence the predictive models and consequently, the results. Extracting a right feature set can boost the performance of a model.
4. **Data modelling:** After extracting the relevant data set and feature set, the next step is to perform the core ML task. Making use of supervised/ unsupervised learning algorithms, we can evaluate similarity and investigate clusters/communities formed to analyse EA models in a different context.
5. **Model evaluation:** The model evaluation mainly supports the find of the best performing model based on suitable performance metrics that represent the data and how well the chosen algorithm will work in the future.
6. **Performance improvement:** Iterating over different graph analytic or ML algorithms and feature set can result in optimised results.

It is essential to know how the optimal solution for a particular scenario looks like, in order to evaluate different algorithms. Therefore, we sketched an airport departure system, which depicts the functionality of the passengers before boarding an aircraft. This case study strategy will provide a richer understanding of the context for the people and the researcher involved. Further, it is used for the evaluation of all algorithms presented in the following.

This example is modelled as an EA model based on ArchiMate 3.0.1. The model incorporates the ArchiMate layers business, application, and technology. It consists of 171 different elements and 250 relations. An excerpt of the model is presented in Figure 1.

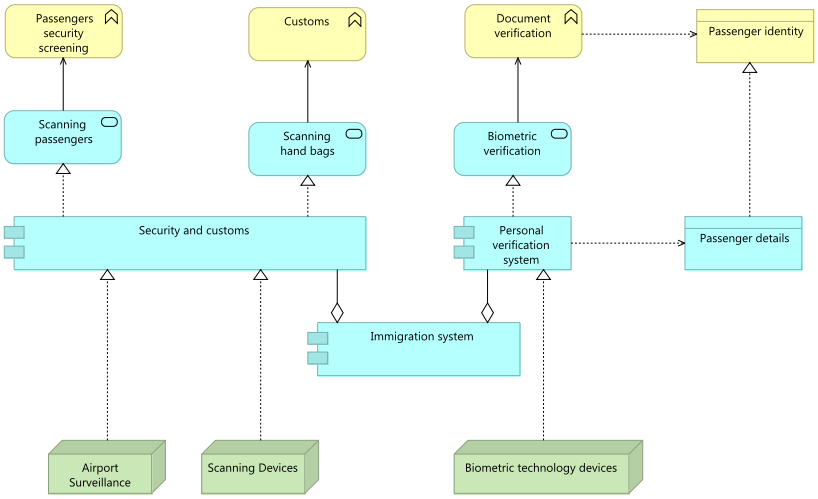


Fig. 1: An Excerpt of the Immigration Process' Application Layer.

4 Graph Analytic Approaches

We depict the EA model as a graph representing ArchiMate components as nodes and relationships between the components as edges. Thus, analysing EA model as graph provides a deeper understanding. As there are several graph analytic techniques, we focus on graph-based similarity and community detection (unsupervised learning).

4.1 Similarity Between Enterprise Architecture Models

Considering EA model as a graph like structure, we apply graph similarity techniques to assess the match between EA models. Graph similarity involves determining a degree of similarity between two graphs [16] quantified by a similarity score between 0 (no similarity) and 1 (complete similarity).

Computing node similarity Node matching similarity is based on examining the content of the components name. Therefore, we compare EA models to each other, where the naming of the components differs only slightly like having typing errors, but both models have the same expression.

Syntactic Similarity. The syntactic approach is related to both, the occurrence of terms and the number of words in the text. In this paper, we use the Cosine similarity index [3] and Jaccard similarity index [20] as a syntactic approach to detect the match between components of two EA models. Those two algorithms serve as starting point for our research and will be extended in future by other algorithms like Levenshtein, N-Gram, Q-Gram, or Sørensen-Dice coefficient.

To find the Cosine and Jaccard similarity score between the phrases (component names) present in two EA models, phrases are turned into words, words are then converted to binary vectors.

Apart from the cosine similarity, another well-known measure for determining the degree of similarity is the Jaccard similarity index. The Jaccard similarity index measures the similarity between finite sample sets and is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets [13].

Latent Semantic Analysis. An alternative method to compare the similarity between EA models is based on fetching the relations between words in the texts. LSA [17] is a corpus-based method which does not use the semantic network, grammar, syntax, and dictionaries. LSA applies a statistical analysis of the latent structures of the documents by finding the underlying meaning or concepts between the documents. It tries to map the words in a document into a concept space and then comparing within that space.

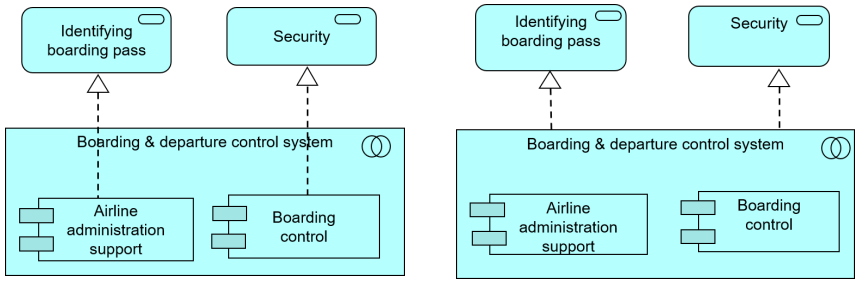
Computing edge similarity The second similarity metric uses the structure of the EA model. This method is applicable when the node correspondence is known and the edges do not vary significantly. The main idea behind this approach is: a node in one graph is said to be similar to a node in another graph if they share a standard set of neighbourhoods. This works recursively [26].

Architect 1 models the application layer, as shown in Figure 2a, which shows the presence of the realisation relationship between Airline administration support (Application Component) and Identifying boarding pass (Application Service) and between Boarding control (Application Component) and Security (Application Service). Architect 2 models a similar kind of application layer, as shown in Figure 2b. Both models have an equal number of components with varying edge connections. Since both the application components have collaborated within Boarding & departure control system (Application Collaboration), there is a similarity between the models generated by both the architects irrespective of edge connection.

Motivated from the work of Liben-Nowell and Kleinberg [18], we selected the most widely used structural similarity measures based on edges, namely Jaccard [18], Dice [30], and Adamic-Adar [1] similarity indexes.

4.2 Nearest Neighbour

The nearest neighbour approach [26] can provide a graph-based recommendation for enterprise architects in order to avoid modelling already existing elements. Early work on this domain was attempted by Aier and Schnherr [2]. Thus, finding communities on a network helps to investigate the roles in an EA model where components in a single cluster will have similar kind of roles compared to other clusters [25].



(a) Application View of $Model_1$ (b) Application View of $Model_2$

Fig. 2: Two simple instances illustrating structural similarity.

We present a technique based on unsupervised learning method to determine the communities [11]. Considering an EA model as a graph [27], we apply different community detection algorithms and analyse the results [24]. The community detection method tries to group a set of vertices having a higher probability of being interconnected than being connected to the members of other groups. In this way, it is possible to investigate similar kinds of grouped components.

Here, similarity refers to the components having a common subset of neighbours. Identifying communities in an EA model provides decision support for the enterprise architects to place new components into the model repository based on identified roles and position of the components. When the size of the model is complex enough to analyse EA model, community detection technique becomes robust regarding execution speed and outperforms SimRank approach, which is discussed by Borozanov et al. [4].

Once a community detection algorithm is implemented and the network is partitioned into communities, it is important to interpret the results to know which algorithm performs best and detect meaningful communities. Since we do not know the gold standard to which the communities should belong, an effective approach to evaluate formed communities is by using internal criteria such as "modularity" [19].

5 Results

In this section, we outline the evaluation results to select the best performing similarity method and the best performing community detection algorithm that provides automatic view generation of EA models.

5.1 Similarity Between Enterprise Architecture Models

Node Similarity Evaluation

Syntactic Similarity. Concerning our example from section 4.1, Cosine and Jaccard similarity measures are applied to assess the syntactic similarity between component names of the EA models. The comparison results for our case shows that cosine similarity (0.700) has a better similarity score which is closer to value 1 than the Jaccard Index (0.537). Thus, we conclude cosine similarity performs better than the Jaccard index in capturing the similarity.

Latent Semantic Analysis. Knowing that the models from section 4.1 convey similar meaning (and from the results presented above), we conclude that by using LSA (0.832) we got the highest score closer to 1. Thus, LSA outperforms Cosine and Jaccard similarity measure in capturing relatedness between EA models. Therefore, any one of the models can be added to the EA repository. Thus, LSA technique is more efficient in finding similarity between EA models compared to simple cosine similarity (0.700) based on the type of content present between the component names. For a certain threshold (similarity score) closer to 1, enterprise architects can decide upon adding a new model into the repository.

Structural similarity evaluation In order to assess the structural similarity between two graphs, we first calculate the similarity between the pair of vertices by constructing similarity matrices for the graphs using Jaccard, Dice, and Adamic-Adar indexes.

Association or correlation between two similarity matrices is later computed by testing correlation to compare whether two graphs are similarly based on its structure. The correlation coefficient can range in value from -1 to $+1$. The larger the absolute value of the coefficient, the stronger the relationship between the variables. We used the Pearson correlation coefficient to compute the association between two similarity matrices.

Our results show that best associations were obtained using the Jaccard similarity coefficient (0.968) whose correlation score is closer to 1 than the Dice similarity coefficient (0.956) and the Adamic-Adar similarity coefficient (0.853). Thus, we choose the Jaccard similarity index as the best performing metric in finding the structural similarity.

5.2 Comparison Between Community Detection Algorithms

In this section, we discuss the approach to find the best-performing community detection algorithm. Since we do not know to which the communities should belong, we use modularity score (internal measure) as a general criterion to find the best-performing algorithm. We consider two different cases in determining the communities of similar kind of EA components. Thus, for each case, we evaluate the performance of different community detection algorithms.

In the following, we consider that the graph does not contain isolated nodes. Table 1 summarises the resulting community size, modularity score, execution time, and ranking of different community detection algorithms.

As shown in Table 1, spinglass takes a top position. The higher the modularity score, the better the network partition. Leading eigenvector and walktrap

Algorithm	Size	Modularity	Running time(in sec)
Edge-betweenness	27	0.579	≈0.05
Infomap	28	0.694	≈0.08
Label propagation	26	0.601	≈50
Leading eigenvector	14	0.749	≈0.11
Spinglass	14	0.786	≈7.82
walktrap	18	0.746	≈0.009

Table 1: Modularity evaluation result

returns the almost equal result with modularity score of 0.75. In this situation, running time, t, is considered as a tie-break criterion. Since walktrap computes faster than leading eigenvector, the user can decide between the selection of an algorithm based on the running time of the specific algorithm. Edge-betweenness algorithm performed worst in this case with lowest modularity score of about 0.579. The lesser the modularity score indicates, the larger the number of nodes in a single community, which makes it non-trivial to understand the distribution of communities formed within a network.

6 Conclusion

With the goal of supporting EA model maintenance, this paper presented a performance-based comparison between different graph similarity and community detection algorithms as candidates in providing a better user experience and suitability in a working environment. The presented approaches enable enterprise architects to track changes in EA models and to avoid EA repository pollution by adding duplicate components.

One limitation of finding similarity method is that computation time is linear concerning the product of graph sizes due to the size of the similarity matrix. A further challenge is to determine the similarity between networks and to define a measure of similarity. For instance, it would be difficult for a domain expert to choose a threshold on the similarity measures to decide whether two models are similar. Another limitation is that the igraph community detection algorithm fails to handle overlapping communities.

Moreover, abstracting EA models as graphs leads to loss of information as it means a focus on the structural facet of the EA model. However, enterprise architects also encode information by grouping elements visually to each other. This information gets lost within our transformation into a graph. Lastly, we focused on finding similarities based on the component names. As future work, additional similarity approaches can be extended to find similarity between EA models based on the description information of an EA component. Furthermore, applying other graph similarity measures may be useful to identify duplicate components in EA models (e.g., [31]).

References

1. Adamic, L.A., Adar, E.: Friends and neighbors on the web. Social networks **25**(3), 211–230 (2003)
2. Aier, S., Schnherr, M.: Integrating an enterprise architecture using domain clustering (2007)
3. Alhadi, A.C., Deraman, A., Yussof, W.N.J.W., Mohamed, A.A., et al.: An ensemble similarity model for short text retrieval. In: International Conference on Computational Science and Its Applications. pp. 20–29. Springer (2017)
4. Borozanov, V., Hacks, S., Silva, N.: Using machine learning techniques for evaluating the similarity of enterprise architecture models - technical paper. In: Advanced Information Systems Engineering - 31st International Conference, CAiSE. pp. 563–578 (2019)
5. Csardi, G., Nepusz, T.: The igraph software package for complex network research. InterJournal, Complex Systems **1695**(5), 1–9 (2006)
6. Dijkman, R., Dumas, M., Van Dongen, B., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Information Systems **36**(2), 498–516 (2011)
7. Dreyfus, D., Iyer, B.: Enterprise architecture: A social network perspective. In: HICSS'06. Proceedings of the 39th Annual Hawaii International Conference on System Sciences, 2006. vol. 8. IEEE (2006)
8. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Automation processes for enterprise architecture management. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops. pp. 340–349. IEEE (2011)
9. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., Hanschke, I.: Requirements for automated enterprise architecture model maintenance. In: 13th International Conference on Enterprise Information Systems (ICEIS), Beijing (2011)
10. Farwick, M., Pasquazzo, W., Breu, R., Schweda, C.M., Voges, K., Hanschke, I.: A meta-model for automated enterprise architecture model maintenance. In: 2012 IEEE 16th International Enterprise Distributed Object Computing Conference. pp. 1–10. IEEE (2012)
11. Fortunato, S.: Community detection in graphs. Physics reports **486**(3-5), 75–174 (2010)
12. Ganser, A., Lichter, H., Roth, A., Rumpe, B.: Staged model evolution and proactive quality guidance for model libraries. Software Quality Journal **24**(3), 675–708 (Sep 2016)
13. Jain, A., Jain, A., Chauhan, N., Singh, V., Thakur, N.: Information retrieval using cosine and jaccard similarity measures in vector space model. International Journal of Computer Applications **164**(6) (2017)
14. Jonkers, H., Lankhorst, M.M., ter Doest, H.W., Arbab, F., Bosma, H., Wieringa, R.J.: Enterprise architecture: Management tool and blueprint for the organisation. Information systems frontiers **8**(2), 63–66 (2006)
15. Kaisler, S.H., Armour, F., Valivullah, M.: Enterprise architecting: Critical problems. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences. pp. 224b–224b. IEEE (2005)
16. Koutra, D., Parikh, A., Ramdas, A., Xiang, J.: Algorithms for graph similarity and subgraph matching. In: Proc. Ecol. Inference Conf. (2011)
17. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse processes **25**(2-3), 259–284 (1998)

18. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. journal of the Association for Information Science and Technology **58**(7), 1019–1031 (2007)
19. Newman, M.E.: Modularity and community structure in networks. Proceedings of the national academy of sciences **103**(23), 8577–8582 (2006)
20. Niwattanakul, S., Singthongchai, J., Naenudorn, E., Wanapu, S.: Using of jaccard coefficient for keywords similarity. In: Proceedings of the International MultiConference of Engineers and Computer Scientists. vol. 1 (2013)
21. O'Neill, T.: Extreme connectivity: EAs need enterprise graphs. `https://www.infoworld.com/article/3045210/enterprise-architecture/extreme-connectivity-eas-need-enterprise-graphs.html/` (2016)
22. Orman, G.K., Labatut, V.: A comparison of community detection algorithms on artificial networks. In: International Conference on Discovery Science. pp. 242–256. Springer (2009)
23. Orman, G.K., Labatut, V., Cherifi, H.: Qualitative comparison of community detection algorithms. In: International conference on digital information and communication technology and its applications. pp. 265–279. Springer (2011)
24. Orman, G.K., Labatut, V., Cherifi, H.: Comparative evaluation of community detection algorithms: a topological approach. Journal of Statistical Mechanics: Theory and Experiment **2012**(08) (2012)
25. Porter, M.A., Onnela, J.P., Mucha, P.J.: Communities in networks. Notices of the AMS **56**(9), 1082–1097 (2009)
26. Saha, S., Ghrera, S.: Nearest neighbor search in complex network for community detection. arXiv preprint arXiv:1511.07210 (2015)
27. Santana, A., Souza, A., Simon, D., Fischbach, K., De Moura, H.: Network science applied to enterprise architecture analysis: Towards the foundational concepts. In: 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC). pp. 10–19. IEEE (2017)
28. Sapp, C.E.: Preparing and Architecting for Machine Learning. `https://www.gartner.com/binaries/content/assets/events/keywords/catalyst/catus8/preparing_and_architecting_for_machine_learning.pdf/` (2017)
29. Simon, D., Fischbach, K.: It landscape management using network analysis. In: Enterprise Information Systems of the Future, pp. 18–34. Springer (2013)
30. Sørensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on danish commons. Biol. Skr. **5**, 1–34 (1948)
31. Yan, X., Yu, P.S., Han, J.: Substructure similarity search in graph databases. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 766–777. ACM (2005)
32. Yang, Z., Algesheimer, R., Tessone, C.J.: A comparative analysis of community detection algorithms on artificial networks. Scientific Reports **6** (2016)