UC San Diego

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

Graph-based Text Representations:
Boosting Text Mining, NLP and Information Retrieval with Graphs

Fragkiskos D. Malliaros | UC San Diego

Michalis Vazirgiannis | École Polytechnique

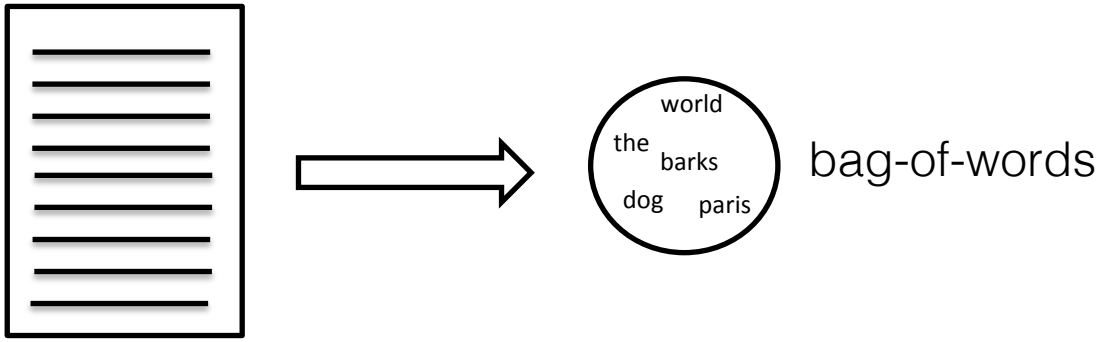
Updated slides at: http://fragkiskosm.github.io/projects/graph_text_tutorial

Conference on Empirical Methods in Natural Language Processing (EMNLP)
Copenhagen, Denmark
September 7–11, 2017

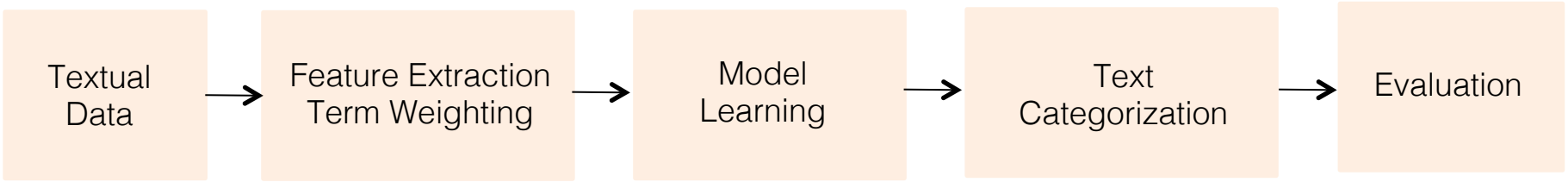
Text Mining – Terminology

- Text mining on a collection of documents:
  - The collection is the data set
  - The documents are the data points

- Since text is unstructured, a document is usually converted in a common representation



Example: Text Categorization



Applications:

- Opinion mining (sentiment analysis)
- Email spam classification
- Web-pages classification
- …

Bag-of-Words (BoW) - Issues

Example text

information retrieval is the activity of obtaining

information resources relevant to an information need

from a collection of information resources

Bag of words: [(activity,1), (collection,1)
(information,4), (relevant,1),
(resources, 2), (retrieval, 1), …]

- Term independence assumption
- Term frequency weighting

Assumptions made by the BoW model

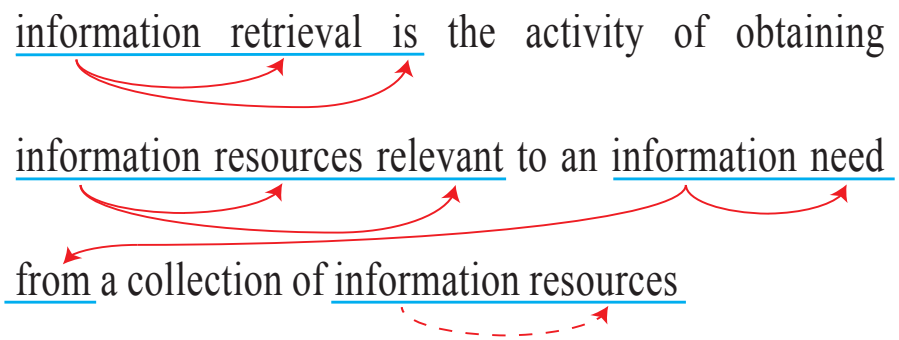
Graph-based Document Representation

- Challenge the term independence and term frequency weighting assumptions taking into account word dependence, order and distance

- Employ a graph-based document representation capturing the above

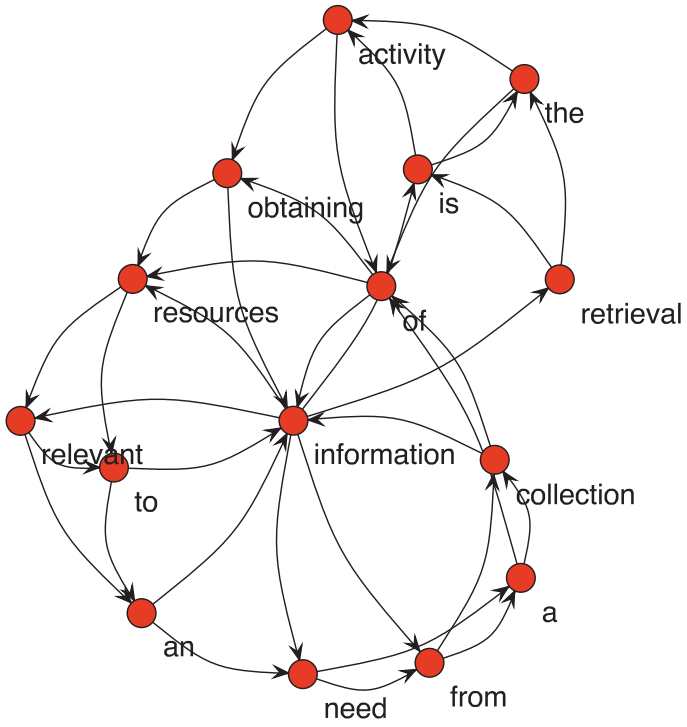
- Graphs have been successfully used in IR to encompass relations and propose meaningful weights (e.g., PageRank)

Graph-based Document Representation - Example



Idea: Replace term frequency with node centrality

Captures: frequency, order and distance



Goal of the Tutorial and Outline

Goal: offer a comprehensive presentation of recent methods that rely on graph-based text representations to deal with various tasks in NLP and IR

- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

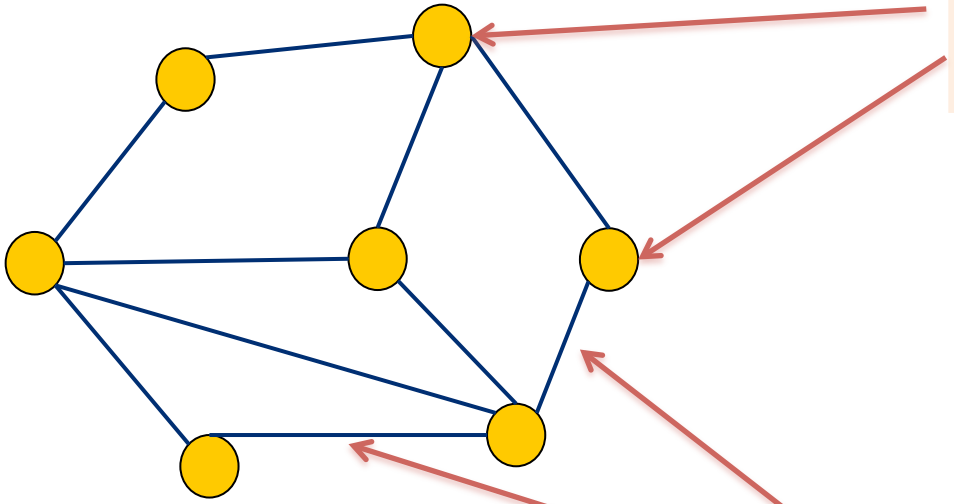
Tutorial Outline

- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

Basic graph-theoretic concepts and definitions

Graphs and Networks

Graphs: modeling dependencies



Nodes (or vertices)
(objects/entities)

Edges (or links)
(interconnections)

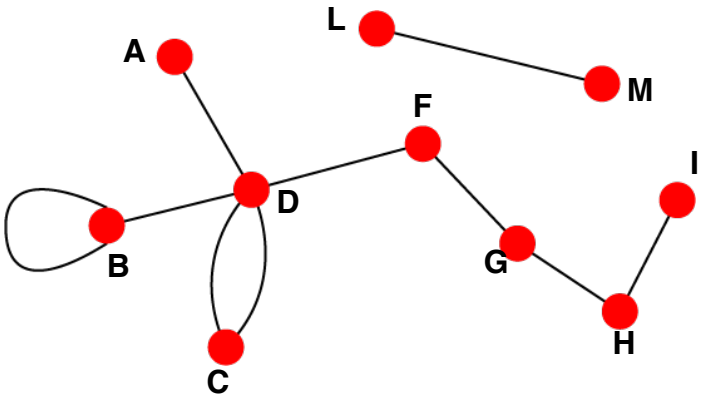
$G = (V, E)$
(network or graph)

$n = |V|$ is the number of nodes
$m = |E|$ is the number of edges

Undirected vs. Directed Networks

Undirected

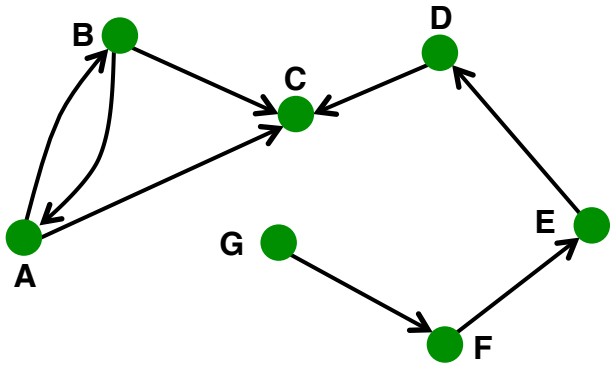
- **Links:** undirected (symmetrical, reciprocal)



- Examples
  - Collaborations
  - Friendship on Facebook

Directed

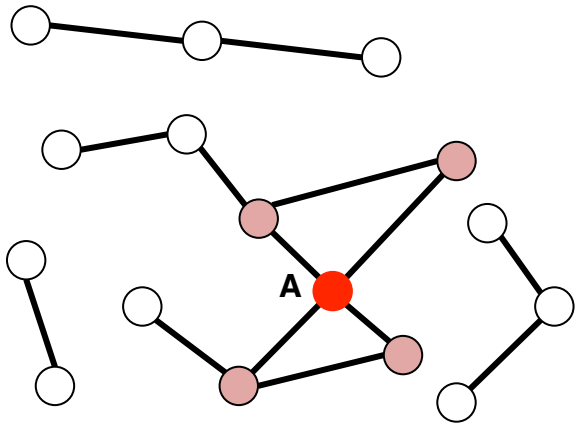
- **Links:** directed (arcs)



- Examples
  - Phone calls
  - Following on Twitter

Node Degree

Undirected



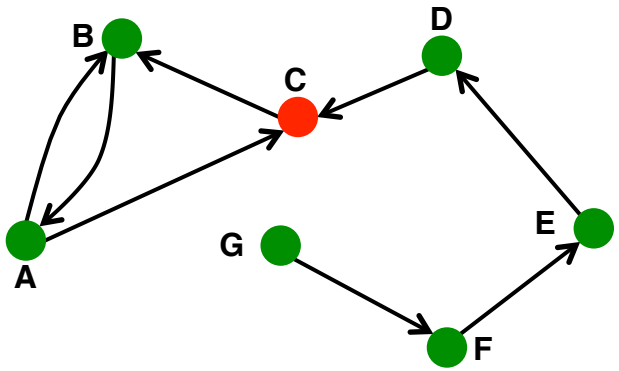
Node degree k_i: the number of edges adjacent to node i

$$k_A = 4$$

Average degree:

$$\bar{k} = \langle k \rangle = \frac{1}{n} \sum_{i=1}^{n} k_i = \frac{2|E|}{n}$$

Directed



In directed networks we define an in-degree and out-degree

The (total) degree of a node is the sum of in- and out-degrees

$$k_C^{in} = 2 \qquad k_C^{out} = 1 \qquad k_C = 3$$

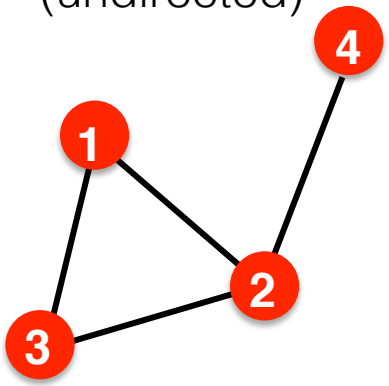
Source: Node with $k^{in} = 0$

Sink: Node with $k^{out} = 0$

Average: $\bar{k^{in}} = \bar{k^{out}}$

More Types of Graphs

Unweighted
(undirected)



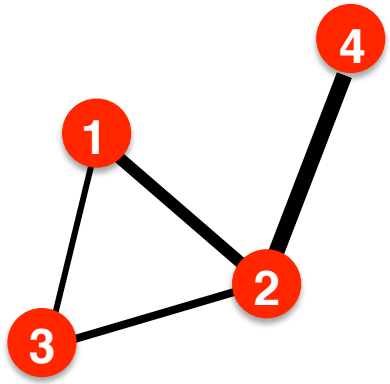
$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$|E| = \frac{1}{2}\sum_{i,j=1}^{n} A_{ij} \qquad \bar{k} = \frac{2|E|}{n}$$

Examples: Friendship, Hyperlink

Weighted
(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

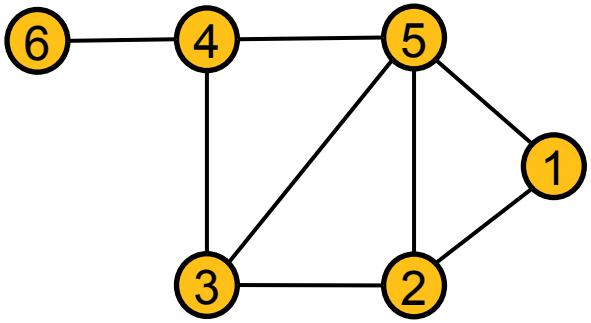
$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$|E| = \frac{1}{2}\sum_{i,j=1}^{n} nonzero(A_{ij}) \qquad \bar{k} = \frac{2|E|}{n}$$

Examples: Collaboration, Internet, Roads

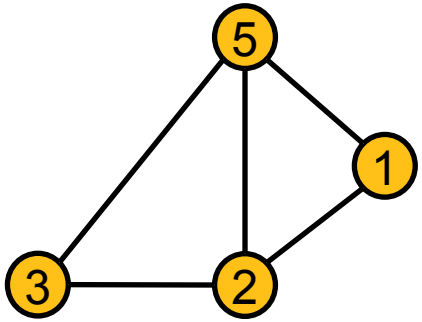
Subgraphs

- Let *G = (V, E)* be a graph and let $S \subseteq V$ be any subset of its vertices



- Definition: The induced subgraph *G[S] = (S, E')* is the graph whose vertex set is *S* and its edge set consists of all of the edges in *E* that have both endpoints in *S*

S={1, 2, 3, 5}



Representation Matters!

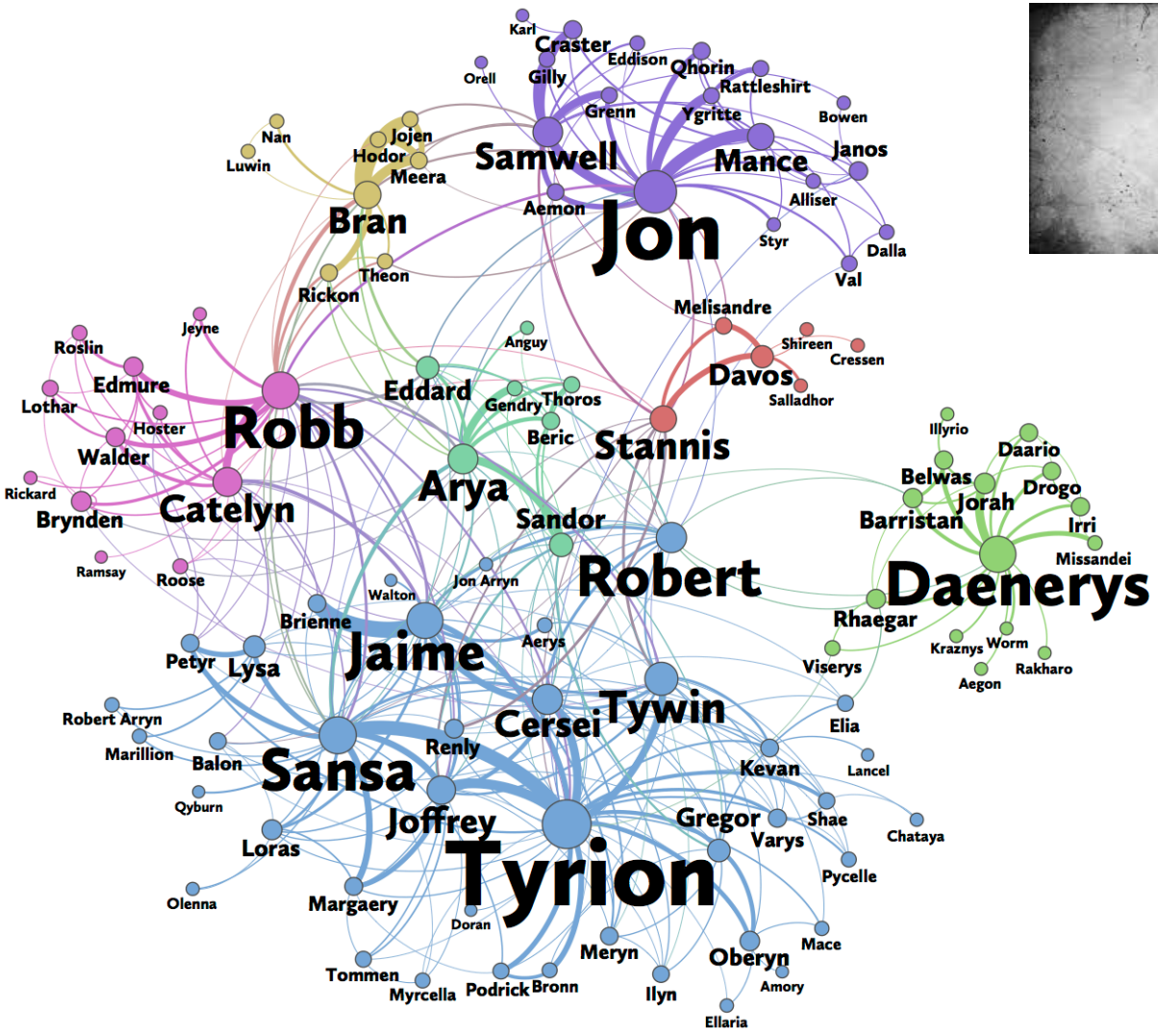
Choice of the proper network representation of a given system determines our ability to use networks successfully

Centrality criteria

Centrality in Networks (1/2)

- Determine the relative importance of a node in the network
  - Applications in Social Network Analysis, the Internet, Epidemiology, Urban informatics, …
- What do we mean by centrality?
  - A central node is more important or powerful …
  - Or, more influential ...
  - Or, is more critical due to its location in the graph
- Also, very closely related to the problem of ranking in the context of Web search
  - Each webpage can be considered as a 'user'
  - Each hyperlink is an endorsement relationship
  - Centrality measures provide a query independent link-based score of importance of a web page

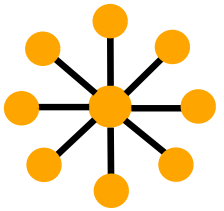
Centrality in Networks (2/2)



Source: https://www.macalester.edu/~abeverid/thrones.html

Types of Centrality

- Starting point: the central node of a star is the most important
- Why?
  - The node with the highest degree
  - The node that is closest to the rest nodes (e.g., has the smallest average distance to other nodes)
  - The node through which all shortest paths pass
  - The node that maximizes the dominant eigenvector (the one that corresponds to the largest eigenvalue) of the adjacency matrix
  - The node with highest probability in the stationary distribution of a random walk on the graph



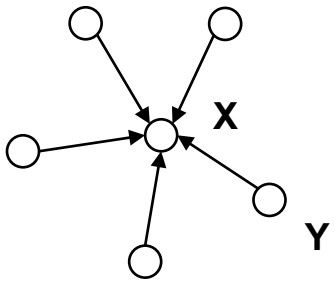
Various competing views of centrality

Measures of Centrality

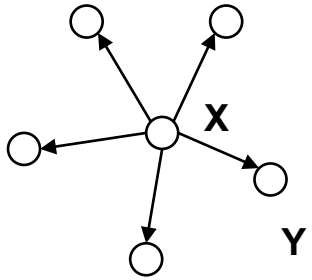
- This observation leads to the following classes of indices of centrality:
  - Measures based on distances (e.g., degree, closeness)
  - Measures based on paths (e.g., betweenness, Katz's index)
  - Spectral measures (eigenvector, PageRank, HITS, SALSA, random walks with restarts)
  - Measure based on groups of nodes (e.g., cliques, plexes, cores)
    - Related to the "clustering" structure
    - More on that in another lecture

A First Example

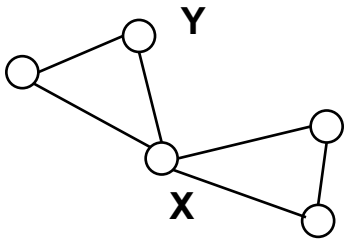
In each of the following networks, X has higher centrality than Y according to a particular measure



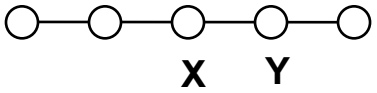
in-degree



out-degree



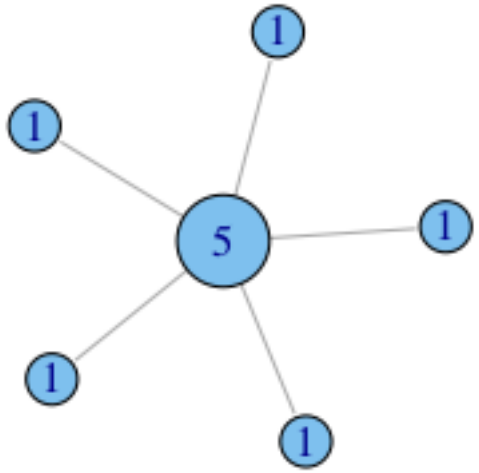
betweenness



closeness

Degree Centrality (1/2)

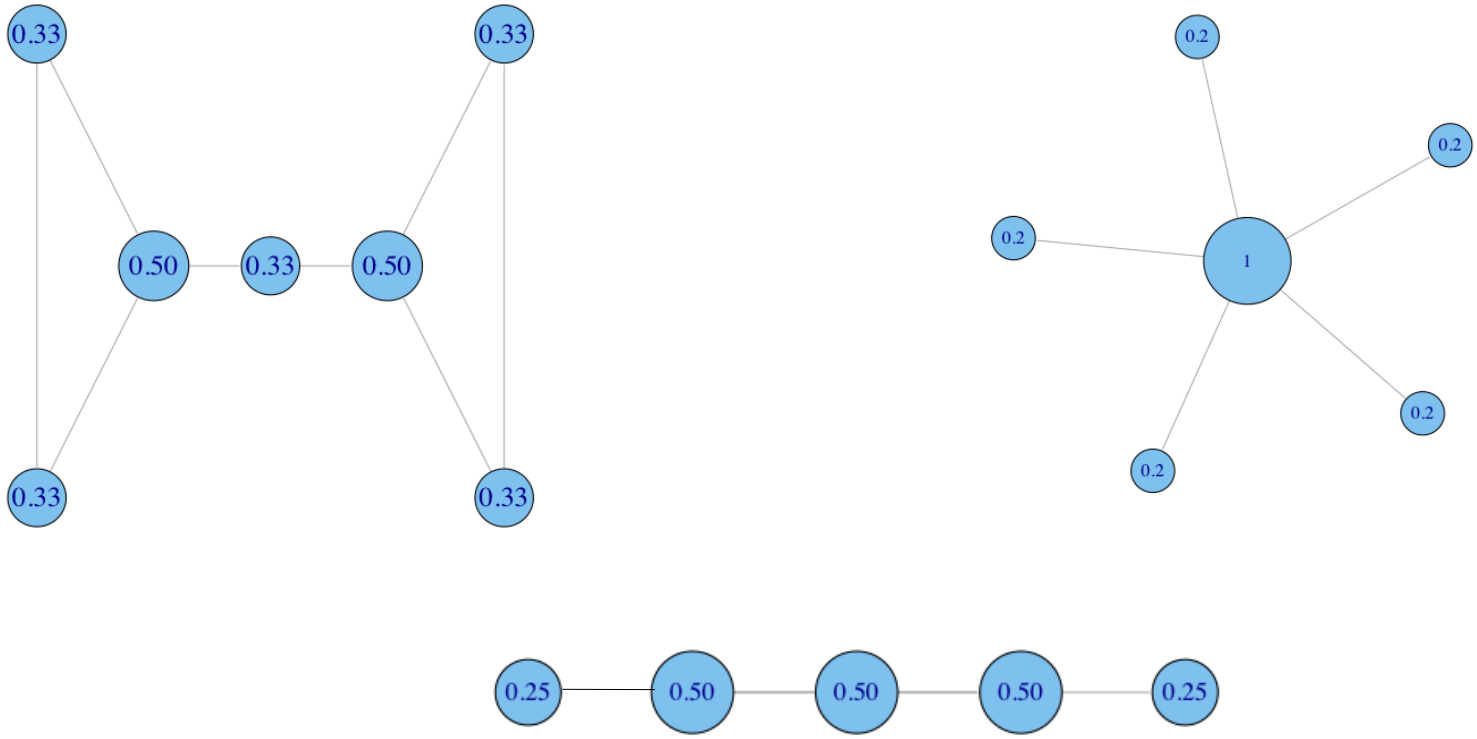
- Idea: A central node is one with many connections



- $C_d\ (i) = k(i)$, where $k(i)$ is the degree of node i

Degree Centrality (2/2)

- Idea: A central node is one with many connections



- Normalized degree centrality: divide by the max possible degree (*n-1*)

Closeness Centrality

- Motivation: it measures the ability to quickly access or pass information through the graph

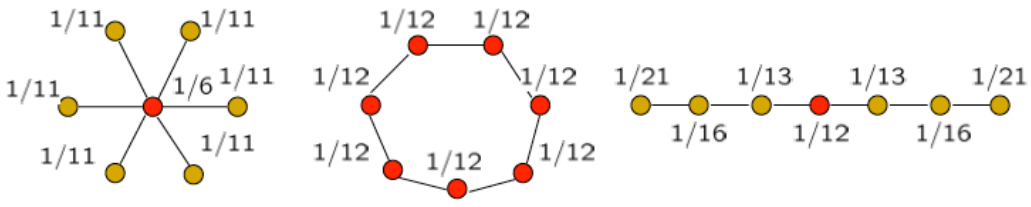
$$C_{Cl}(i) = \frac{n-1}{\sum_{j \neq i} d(i,j)}$$

Values in the range [0,1]

Mean distance from a node to other nodes

d(i, j) is the length of the shortest path between *i* and *j* (geodesic distance)

- The closeness of a node is defined as the inverse of the sum of the shortest path (SP) distances between the node and all other nodes in the graph



Be close to everybody else
(e.g., influence on other nodes)

Why inverse the distance?
- Nodes with low mean distance should get high score

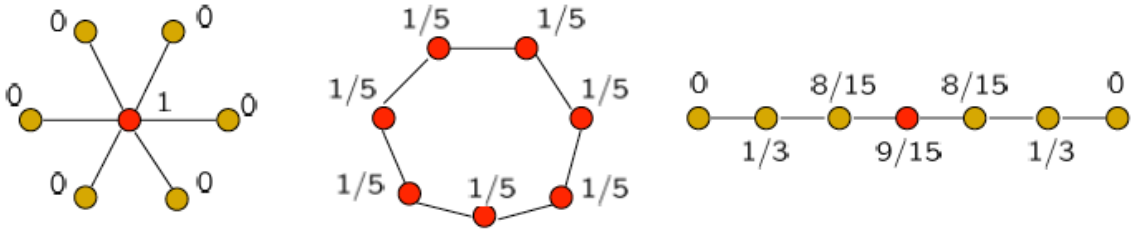
[Mateos, '17]

Betweenness Centrality

- Motivation: a node is important if it lies in many shortest paths

$$C_{bt}(i) = \sum_{s \neq i \neq t \in V} \frac{\sigma(s,t|i)}{\sigma(s,t)}$$

- $\sigma(s, t)$ is the total number of shortest paths from s to t
- $\sigma(s, t|v)$ is the number of shortest paths from s to t that pass through i



Essential nodes in passing information through the network

Oftentimes it is normalized: $\frac{C_{bt}(i)}{\binom{n-1}{2}}$

The HITS Algorithm

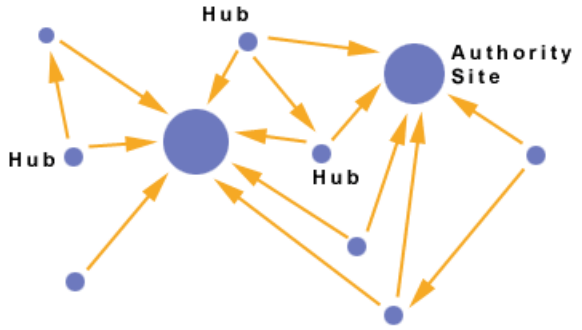
(Hubs and Authorities)

Hubs and Authorities (1/3)

Interesting pages fall into two classes:

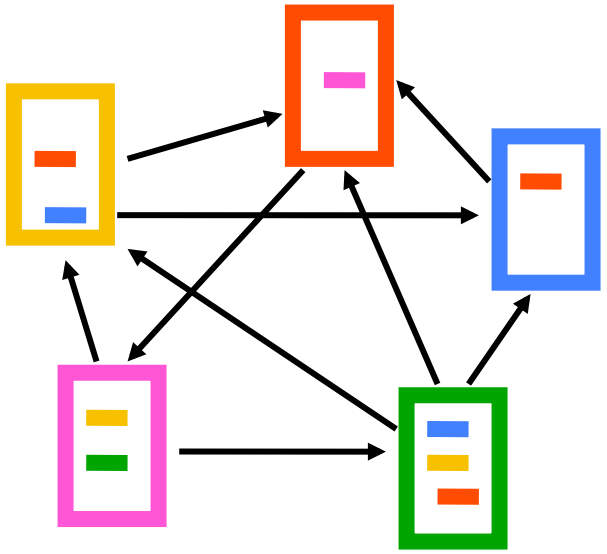
1. Authorities are pages containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of auto manufacturers

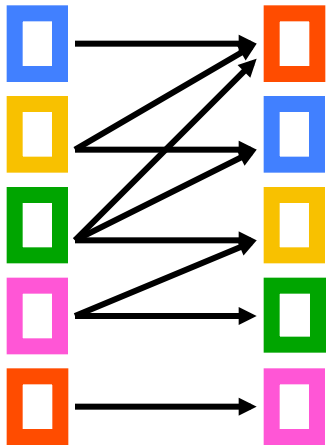
2. Hubs are pages that link to authorities
   - List of newspapers
   - Course bulletin
   - List of U.S. auto manufacturers



Hubs and Authorities (2/3)

- Pages have double identity
  - Hub identity
  - Authority identity
- Good hubs point to good authorities
- Good authorities are pointed by good hubs





Hubs and Authorities (3/3)

- Two kind of weights:
  - Hub weight
  - Authority weight

- The hub weight is the sum of the authority weights of the authorities pointed to by the hub

- The authority weight is the sum of the hub weights that point to this authority

- Represented as vectors h and a, where the i^{th} element is the hub/authority score of the i^{th} node

HITS Algorithm

- Initialize: $a_j^0 = 1/\sqrt{n}, \quad h_j^0 = 1/\sqrt{n}$
- Repeat until convergence
  - Authority: $a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}, \quad \forall i$
  - Hub: $h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}, \quad \forall i$
  - Normalize: $\sum_i \left(a_i^{(t+1)}\right)^2 = 1$ and $\sum_j \left(h_j^{(t+1)}\right)^2 = 1$

[Kleinberg '98]

HITS and Eigenvectors

- HITS in vector notation
  - $\alpha = [\alpha_1, \alpha_2, ..., \alpha_n]^T$ and $h = [h_1, h_2, \ldots, h_n]^T$
- We can rewrite α_i and h_i based on the adjacency matrix

$$h_i = \sum_{i \to j} \alpha_j \quad \text{as} \quad h_i = \sum_j \mathbf{A}_{ij} \cdot \alpha_j$$

- Thus, $h = A\,\alpha$ and $\alpha = A^T h$

- $\alpha^{(t+1)} = A^T h^{(t)}$ and $h^{(t+1)} = A\,\alpha^{(t)}$
- $\alpha^{(t+1)} = A^T A\,\alpha^{(t)}$ and $h^{(t+1)} = AA^T h^{(t)}$

Repeated iterations will converge to the eigenvectors

SVD

- Authority weight vector α : eigenvector of A^TA
- Hub weight vector h : eigenvector of AA^T

The vectors α and h are the singular vectors of A

PageRank

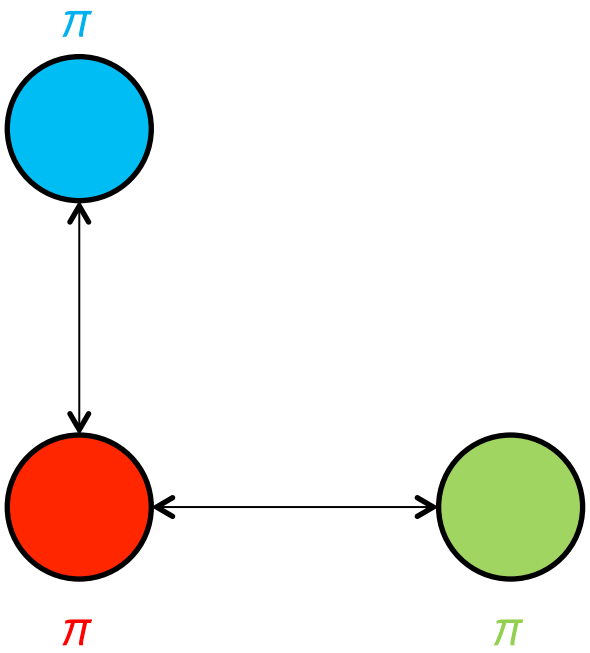
- Good authorities should be pointed by good authorities
  - The value of a node comes from the value of the nodes that point to it

- How do we implement that?
  - Assume that we have a unit of authority to distribute to all nodes
    - Initially, each node gets *1/n* amount of authority
  - Each node distributes its authority value to its neighbors
  - The authority value of each node is the sum of the authority fractions that they collect from their neighbors

$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

π_v: the PageRank value of node *v*
- Recursive definition

A Simple Example



$\pi + \pi + \pi = 1$

$\pi = \pi + \pi$

$\pi = \frac{1}{2}\pi$

$\pi = \frac{1}{2}\pi$

- Solving the system of equations we get the authority values for the nodes
  - $\pi = \frac{1}{2}$ $\pi = \frac{1}{4}$ $\pi = \frac{1}{4}$

A More Complex Example

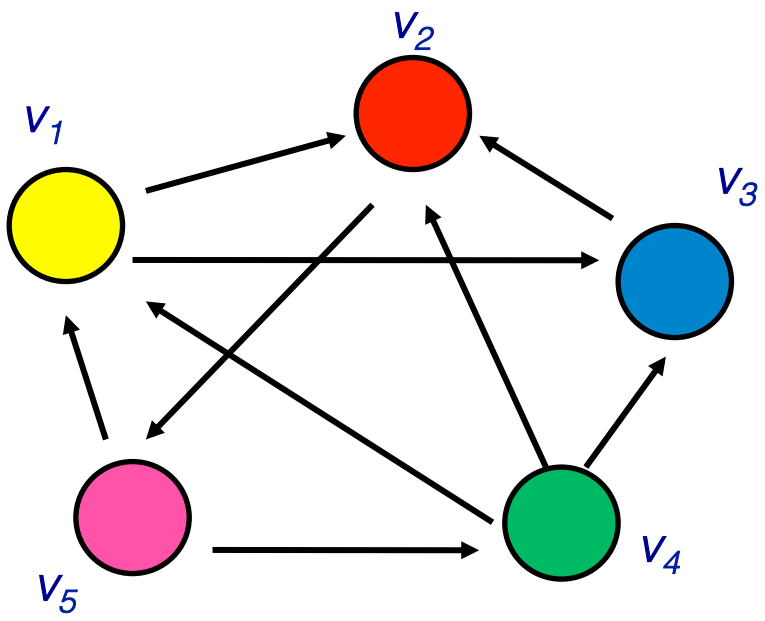
$\pi_1 = 1/3\ \pi_4 + 1/2\ \pi_5$

$\pi_2 = 1/2\ \pi_1 + \pi_3 + 1/3\ \pi_4$

$\pi_3 = 1/2\ \pi_1 + 1/3\ \pi_4$

$\pi_4 = 1/2\ \pi_5$

$\pi_5 = \pi_2$



$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

Computing PageRank Weights

- A simple way to compute the weights is by iteratively updating the weights

Initialize all PageRank weights to *1/n*

Repeat:

$$\pi_v = \sum_{\forall (u,v) \in E} \frac{1}{k_{out}(u)} \pi_u$$

Until the weights do not change

This process converges

Core decomposition in networks

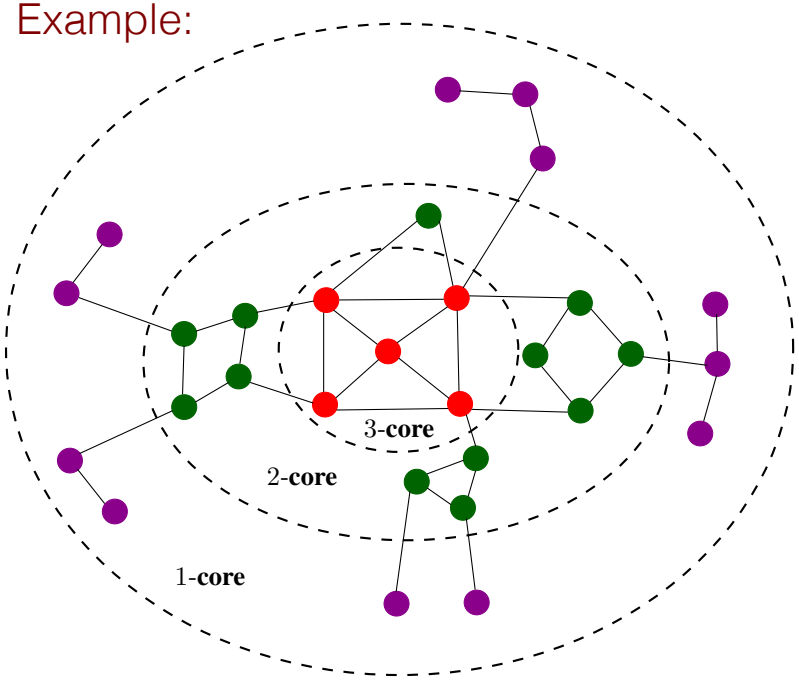
Core Decomposition

- Tool to analyze the structure of real networks
  - Quantify community and clustering structure
- Hierarchical representation of a graph into nested subgraphs of increased connectivity and coherence properties
- Basic idea:
  - Set a threshold on the node degree, say k
  - Nodes that do not satisfy the threshold are removed from the graph
- Extensions to other node properties (e.g., triangles)
- Plethora of applications
  - Dense subgraph discovery and community detection
  - Evaluation of collaboration in social networks
  - Identification of influential spreaders in social networks
  - Text analytics

k-Core Decomposition

- Degeneracy for an undirected graph G
  - Also known as the k-core number
  - The k-core of G is the largest subgraph in which every vertex has degree at least k within the subgraph

Example:



Core number $c_i = 1$

Core number $c_i = 2$

Core number $c_i = 3$

Graph Degeneracy $\delta^*(G) = 3$

$G_0 = G$

$G_1 = 1$-core of G

$G_2 = 2$-core of G

$G_3 = 3$-core of G

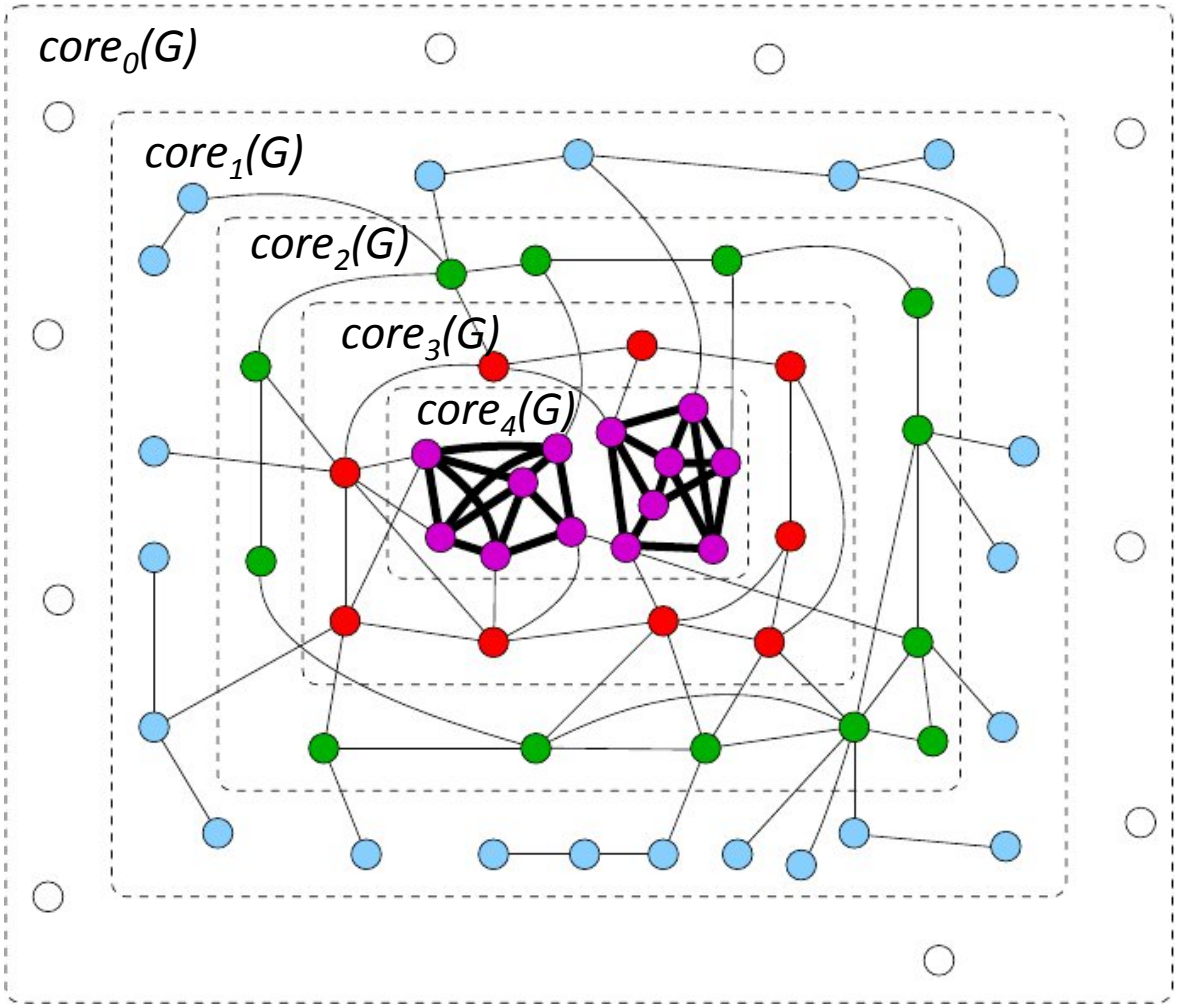
$G_0 \supseteq G_1 \supseteq G_2 \supseteq G_3$

Important property:
- Fast and easy to compute
- Linear to the size of the graph
- Scalable to large scale graphs

Note:
The degeneracy and the size of the k-core provide a good indication of the cohesiveness of the graph

Also known as graph degeneracy

Another Example



Algorithm for k-Core Decomposition

Algorithm k-core(G, k)

Input: An undirected graph G and positive integer k

Output: k-core(G)

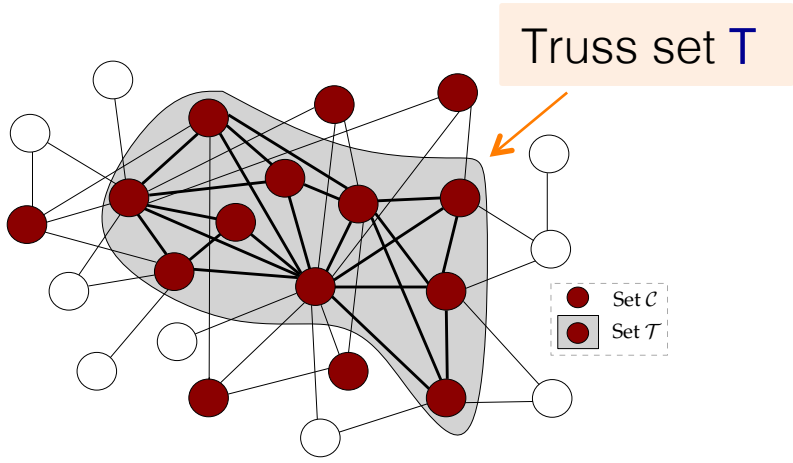
1. let F := G
2. while there is a node x in F such that $\deg_F(x)<k$
   delete node x from F
3. return F

- Many efficient algorithms have been proposed for the computation
  - Time complexity: O(m)

[Batagelj and Zaversnik, '03]

K-truss Decomposition (Triangles)

- K-truss decomposition [Cohen '08], [Wang and Cheng '12]
  - Triangle-based extension of the k-core decomposition
  - Each edge of the K-truss subgraph participates in at least K-2 triangles
    - Informally, the "core" of the maximal k-core subgraph
    - Subgraph of higher coherence compared to the k-core



Graph-based text representations

Graph Semantics

- Let G = (V, E) be the graph that corresponds to a document d
- The nodes can correspond to:
  - Paragraphs
  - Sentences
  - Phrases
  - Words [Main focus of the tutorial]
  - Syllables
- The edges of the graph can capture various types of relationships between two nodes:
  - Co-occurrence within a window over the text [Main focus of the tutorial]
  - Syntactic relationship
  - Semantic relationship

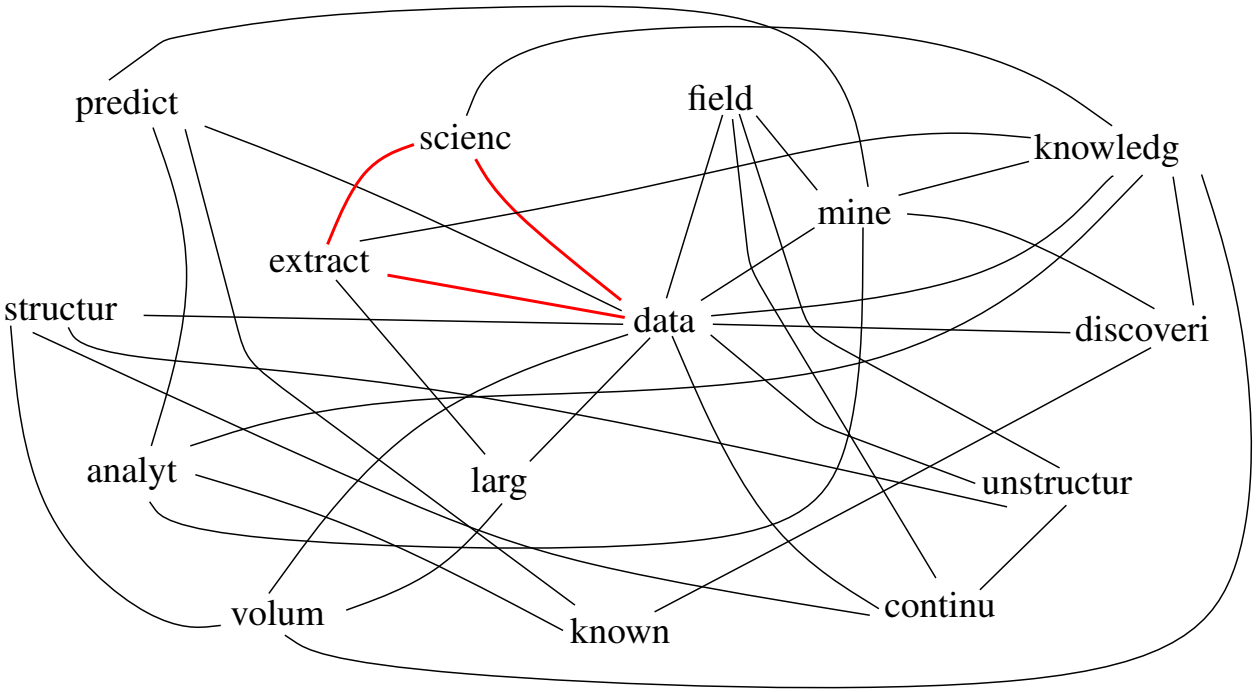
Graph-of-Words (GoW) Model

- Each document $d \in D$ is represented by a graph $G_d = (V_d, E_d)$, where the nodes correspond to the terms t of the document and the edges capture co-occurrence relationships between terms within a fixed-size sliding window of size w
- Directed vs. undirected graph
  - Directed graphs are able to preserve the actual flow of a text
  - In undirected graphs, an edge captures co-occurrence of two terms whatever the respective order between them is
- Weighted vs. unweighted graph
  - The higher the number of co-occurrences of two terms in the document, the higher the weight of the corresponding edge
- Size w of the sliding window
  - Add edges between the terms of the document that co-occur within a sliding window of size w
  - Larger window sizes produce graphs that are relatively dense

[Mihalcea and Tarau, EMNLP '04], [Blanco and Lioma, Inf. Retr. '12],
[Rousseau and Vazirgiannis, CIKM '13]

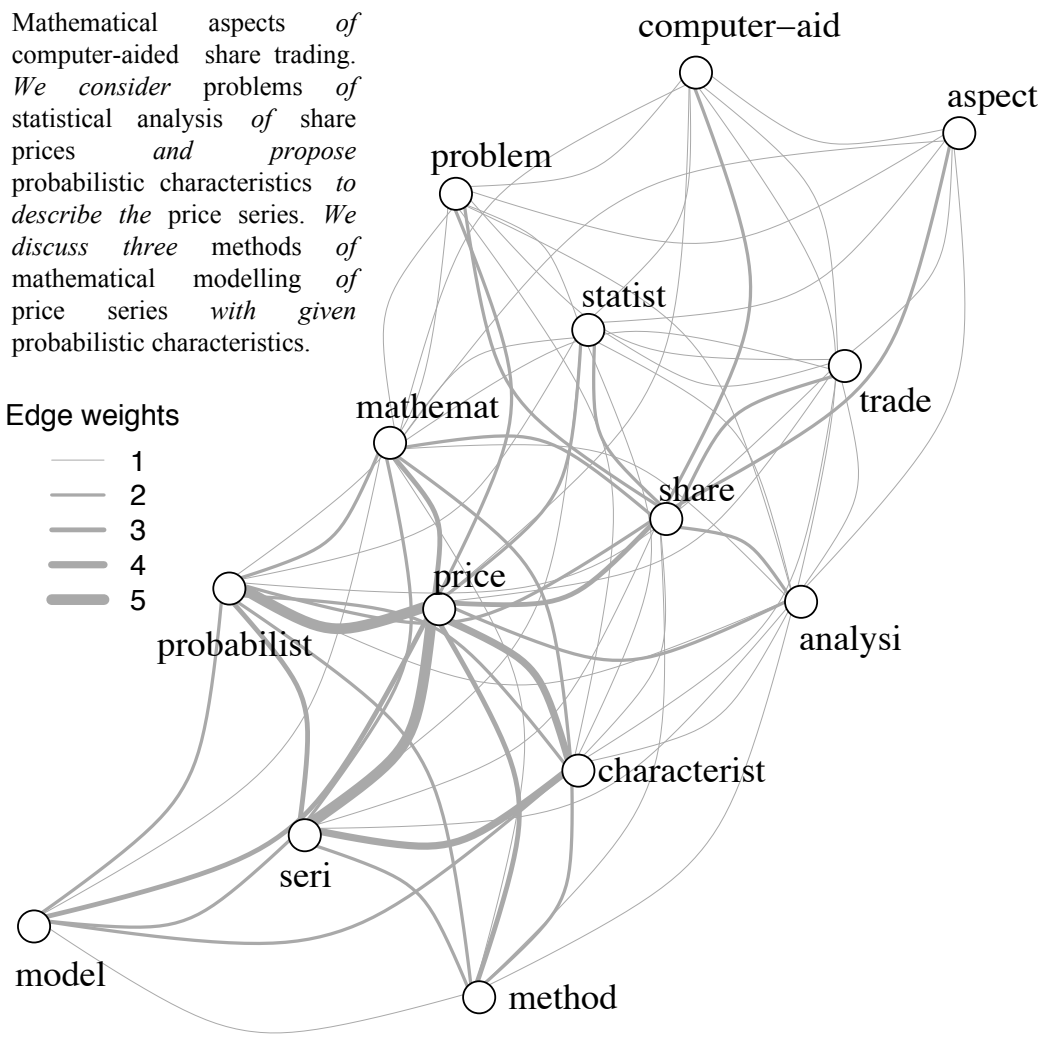
Example of Unweighted GoW

Data Science ~~is the~~ extraction of knowledge from large volumes of data that are structured or unstructured which is a continuation of the field of data mining and predictive analytics, also known as knowledge discovery and data mining.



w = 3
unweighted, undirected graph

Example of Weighed Undirected GoW



Tutorial Outline

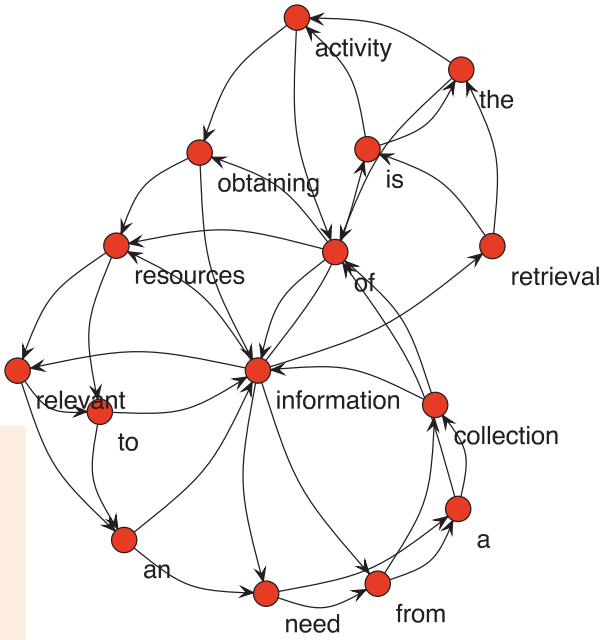
- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

In-degree based TW

- The weight of a term in a document is its in-degree in the graph-of-words
- It represents the number of distinct contexts of occurrence
- We store the document as a vector of weights in the direct index and similarly in the inverted index
- For example:

information	5
retrieval	1
is	2
the	2
activity	2
of	3
obtaining	2
resources	3
relevant	2
to	2
an	2
need	2
from	2
a	2
collection	2

Bag of words:
((activity,1), (collection,1), (information,4), (relevant,1), (resources, 2), (retrieval, 1)..)



TF-IDF and BM25

- Term t, document d, collection of size N, term frequency tf(t, d), document frequency df(t), document length |d|, average document length avdl, asymptotical marginal gain k_1 (1.2), slope parameter b

- TF-IDF [Singhal et al., TREC-7]

$$\text{TF-IDF}(t, d) = \text{TF}_{pol}\text{-IDF}(t, d) = \text{TF}_p\text{oTF}_l(t, d) \times \text{IDF}(t) = \left(\frac{1 + \log\left(1 + \log\left(tf(t,d)\right)\right)}{1 - b + b \times \frac{|d|}{avdl}} \right) \times \log\left(\frac{N+1}{df(t)} \right)$$

- BM25 [Lv and Zhai, CIKM '11]

$$\text{BM25}(t, d) = \left(\frac{(k_1 + 1) \times tf(t,d)}{k_1 \times \left(1 - b + b \times \frac{|d|}{avdl}\right) + tf(t,d)} \right) \times \log\left(\frac{N+1}{df(t)} \right)$$

TW-IDF

- Term t, document d, collection of size N, term weight tw(t, d), document frequency df(t), document length |d|, average document length avdl, asymptotical marginal gain k_1 (1.2), slope parameter b

$$\text{TW-IDF}(t, d) = \left(\frac{tw(t,d)}{1 - b + b \times \frac{|d|}{avdl}} \right) \times \log\left(\frac{N+1}{df(t)} \right)$$

- In the bag-of-word representation, tw is usually defined as the term frequency or sometimes just the presence/absence of a term (binary tf)
- In the graph-of-word representation, tw is the **in-degree** of the vertex representing the term in the graph

[Rousseau and Vazirgiannis, CIKM '13]

Experimental Evaluation

- Datasets
- Platforms
- Evaluation
- Results

Datasets (1/2)

- Disks 1 & 2 (TREC)
  741,856 news articles from Wall Street Journal (1987-1992), Federal Register (1988-1989), Associated Press (1988-1989 and Information from the Computer Select disks (1989-1990)
- Disks 4 & 5 (TREC, minus the Congressional Record)
  528,155 news releases from Federal Register (1994), Financial Times (1991-1994), Foreign Broadcast Information Service (1996) and Los Angeles Times (1989-1990)
- WT10G (TREC)
  1,692,096 crawled pages from a snapshot of the Web in 1997
- .GOV2 (TREC)
  25,205,179 crawled Web pages from .gov sites in early 2004

Datasets (2/2)

Dataset / Statistic	Disks 1 & 2	Disks 4 & 5	WT10G	.GOV2
# of documents	741,856	528,155	1,692,096	25,205,179
# of unique terms	535,001	520,423	3,135,780	15,324,292
average # of terms (avdl)	237	272	398	645
average # of vertices	125	157	165	185
average # of edges	608	734	901	1,185

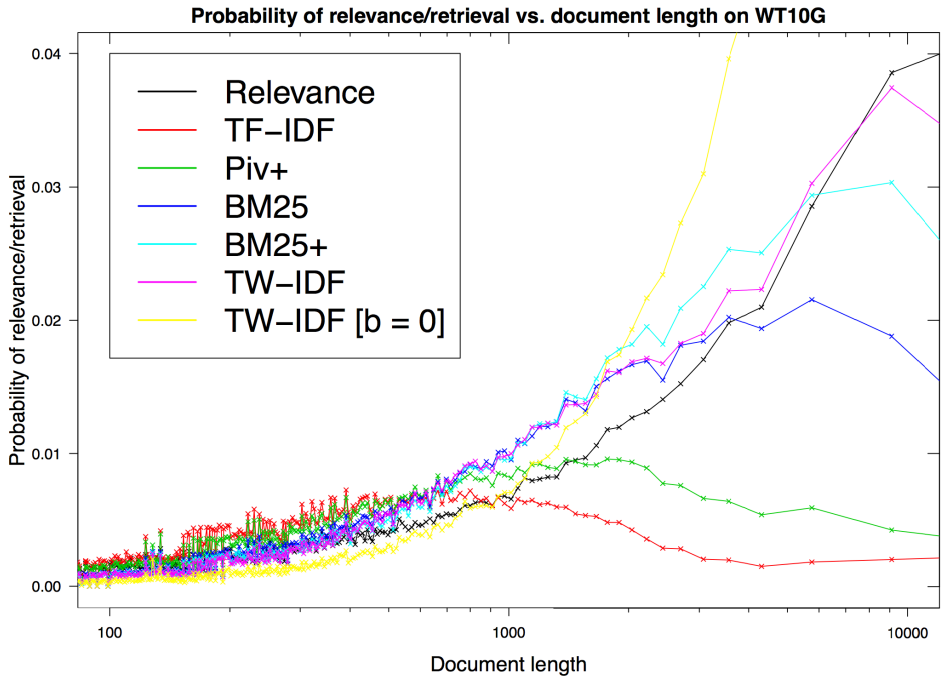
Table: Statistics on the four TREC datasets used; Disks 4&5 excludes the Congressional Record. The average values are computed per document.

Evaluation

- Mean Average Precision (MAP) and Precision at 10 (P@10)
  - Considering only the top-ranked 1000 documents for each run
- Statistical significance of improvement was assessed using the Student's paired t-test
  - R implementation (t.test {stats} package), trec_eval output as input
  - Two-sided p-values less than 0.05 and 0.01 to reject the null hypothesis
- Likelihood of relevance vs. likelihood of retrieval [Singhal et al., SIGIR '96]
- 4 baseline models: TF-IDF, BM25, Piv+ and BM25+
  - Tuned slope parameter *b* for pivoted document length normalization (2-fold cross-validation, odd vs. even topic ids, MAP maximization)
  - Default (1.0) lower-bounding gap [Lv and Zhai, CIKM '11]

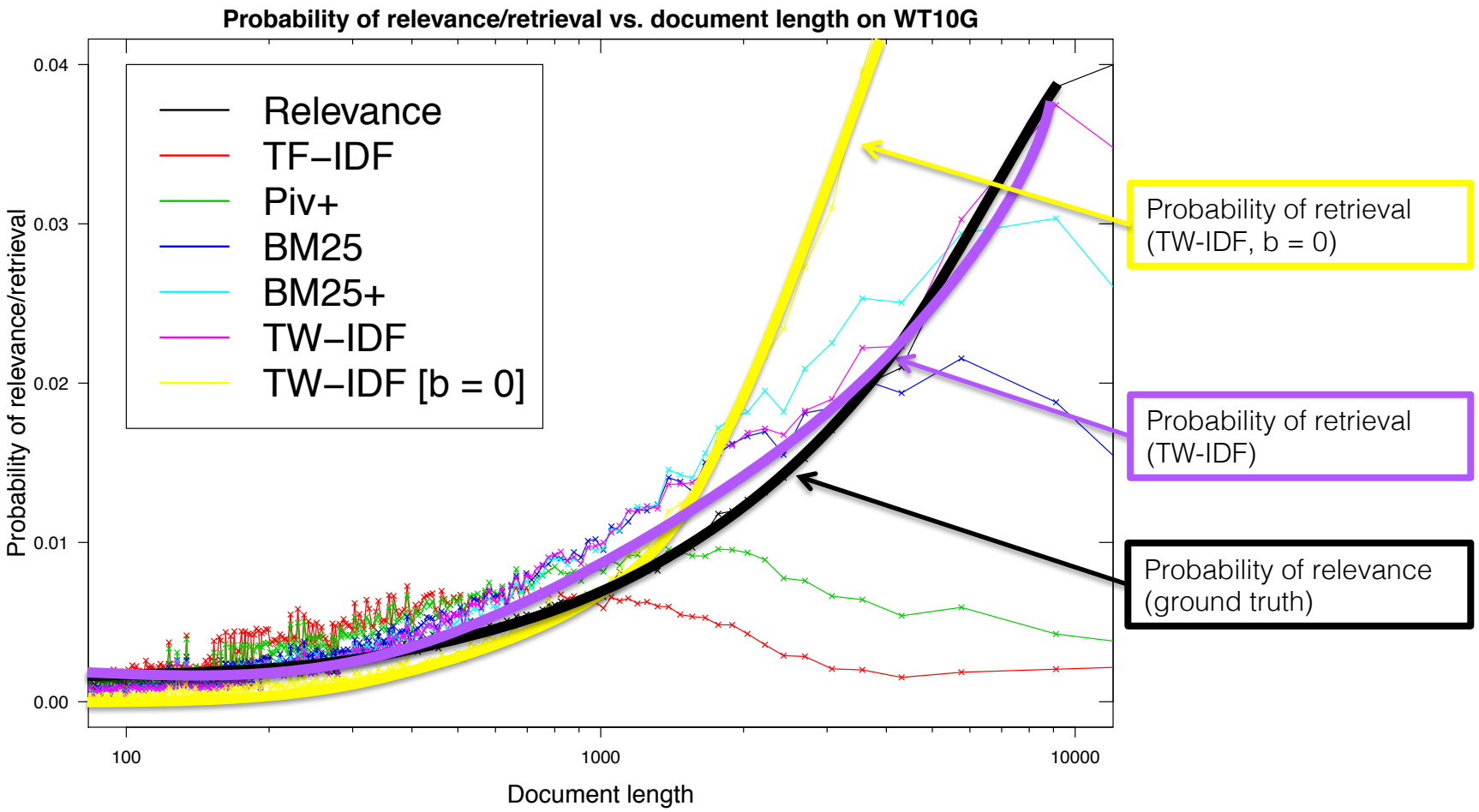
Graph-based Ad Hoc IR

- Evaluation in terms of:
  - Mean Average Precision
  - Precision@10
  - Probability of relevance vs. probability of retrieval



Model	b	TREC1-3 Ad Hoc		TREC 2004 Robust		TREC9-10 Web		TREC 2004-2006 Terabyte	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
TF_{pol}	0.20	0.1471	0.3960	0.1797	0.3647	0.1260	0.1875	0.1853	0.4913
TF_{kop}	0.75	0.1346	0.3533	0.2045	0.3863	0.1702	0.2208	0.2527	0.5342
TW	none	0.1502	0.3662	0.1809	0.3273	0.1430	0.1979	0.2081	0.5021
TW_p	0.003	**0.1576****	**0.4040****	**0.2190****	**0.4133****	**0.1946****	**0.2479****	**0.2828****	**0.5407****
TF-IDF	0.20	0.1832	0.4107	0.2132	0.4064	0.1430	0.2271	0.2068	0.4973
BM25	0.75	0.1660	0.3700	0.2368	0.4161	0.1870	0.2479	0.2738	0.5383
TW-IDF	0.003	**0.1973****	**0.4148***	**0.2403****	**0.4180***	**0.2125****	**0.2917****	**0.3063****	**0.5633****

Likelihood of Relevance vs. Likelihood of Retrieval



Tutorial Outline

- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

Single Document Keyword Extraction

Keywords are used everywhere

- Looking up information on the Web (e.g., via a search engine bar)
- Finding similar posts on a blog (e.g., tag cloud)
- For ads matching (e.g., AdWords' keyword planner)
- For research paper indexing and retrieval (e.g., SpringerLink)
- For research paper reviewer assignment

Applications are numerous

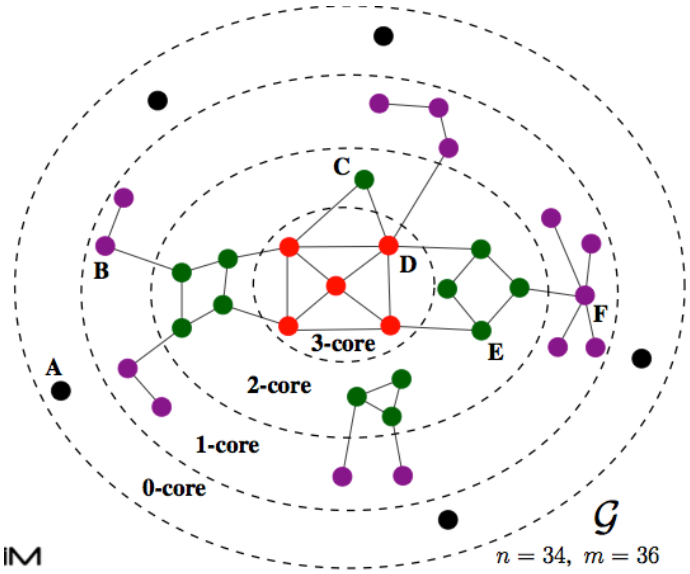
- **Summarization** (to get a gist of the content of a document)
- **Information filtering** (to select specific documents of interest)
- **Indexing** (to answer keyword-based queries)
- **Query expansion** (using additional keywords from top results)

Graph-based Keyword Extraction (1/2)

Existing graph-based keyword extractors:

- Assign a centrality based score to a node
- Top ranked ones will correspond to the most representative

- TextRank (PageRank) [Mihalcea and Tarau, EMNLP '04]
- HITS [Litvak and Last, MMIES '08]
- Node centrality (degree, betweenness, eigenvector) [Boudin, IJNLP '13]

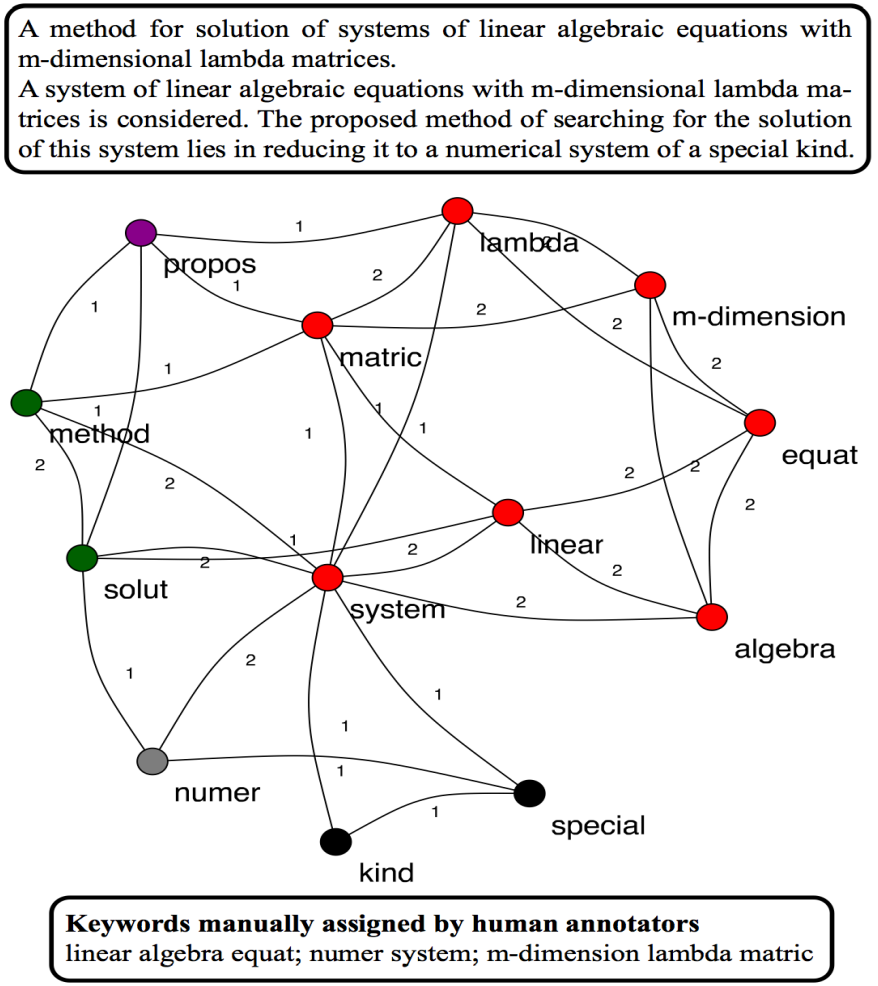


k-core decomposition of the graph

Idea: retain the k-core subgraph of the graph to extract the nodes based on their centrality and cohesiveness

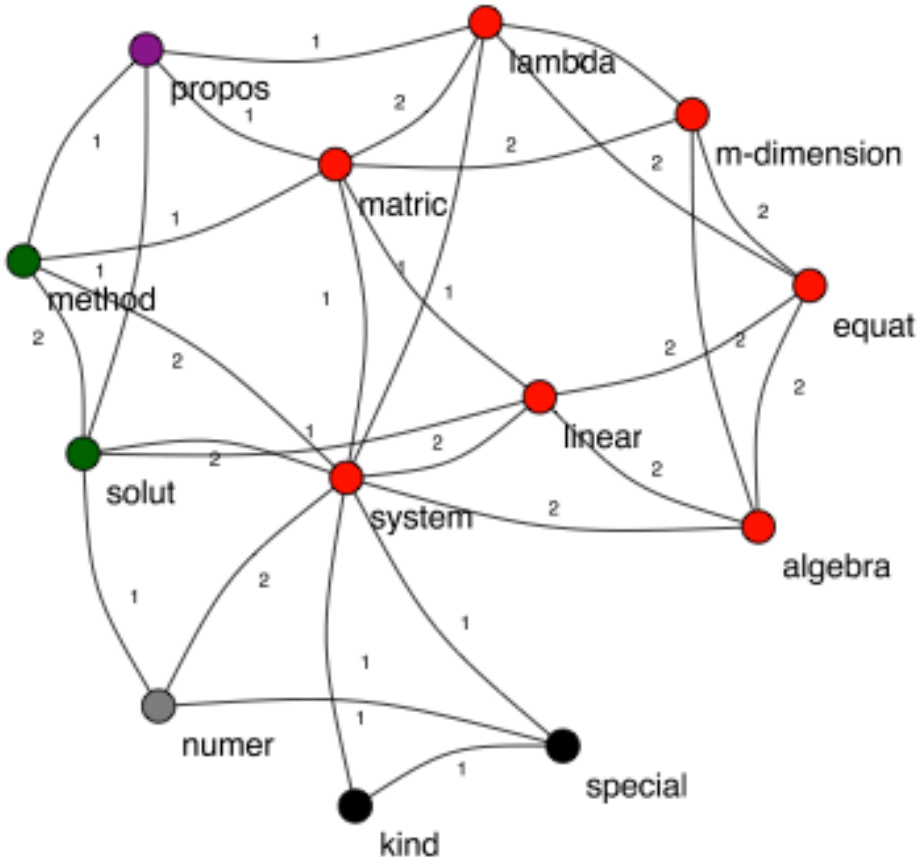
Graph-based Keyword Extraction (2/2)

- Single-document keyword extraction
  - Select the most cohesive sets of words in the graph as keywords
  - Use k-core decomposition to extract the main core of the graph
  - Weighted edges



[Rousseau and Vazirgiannis, ECIR '15]

PageRank vs. k-core



Keywords manually assigned by human annotators
linear algebra equat; numer system; m-dimension lambda matric

WK-core		PageRank	
system	6	**system**	1.93
matric	6	**matric**	1.27
lambda	6	solut	1.10
linear	6	**lambda**	1.08
equat	6	**linear**	1.08
algebra	6	**equat**	0.90
m-dim...	6	**algebra**	0.90
method	5	**m-dim...**	0.90
solut	5	propos	0.89
propos	4	method	0.88
numer	3	special	0.78
specia	2	**numer**	0.74
kind	2	kind	0.55

Keywords are not Unigrams

- 500 abstracts from the *Inspec* database used in our experiments,
- 4,913 keywords manually assigned by human annotators
- only 662 are unigrams (13%).
- Bigrams (2,587 – 52%) … 7-grams (5).

⇒ keywords are bigrams, if not higher order n-grams.

⇒ the interactions within keywords need to be captured in the first place – i.e. in the graph.

⇒ we can consider a k-core to form a "long-distance (k+1)-gram" [Bassiou and Kotropoulos, 2010]

How Many Keywords?

- Most techniques in keyword extraction assign a score to each feature and then take the top ones
- But how many?
  - Absolute number (top X) or relative number (top X%)?
- Besides, at fixed document length, humans may assign more keywords for a document than for another one

X is decided at document level (size of the k-core subgraph)

k-cores are adaptive

Datasets

- *Hulth2003* – 500 abstracts from the *Inspec* database [Hulth, 2003]

- *Krapi2009* – 2,304 ACM full papers in Computer Science (references and captions excluded) [Krapivin et al., 2009]

All approaches are unsupervised and single-document

Models and Baseline Methods

Graph-of-words:

- Undirected edges
- Forward edges
  - Natural flow of the text
  - An edge *term*1 → *term*2 meaning that *term*1 precedes *term*2 in a sliding window
- Backward edges

Keyword extractors:

- PageRank
- HITS (authority scores only)

} Top 33% or top 15% keywords

- k-core
- Weighted k-core

} Main core

Evaluation Metrics

- Each document has a set of golden keywords assigned by humans
  - precision, recall and F1-score per document
  - macro-average each metric at the collection level

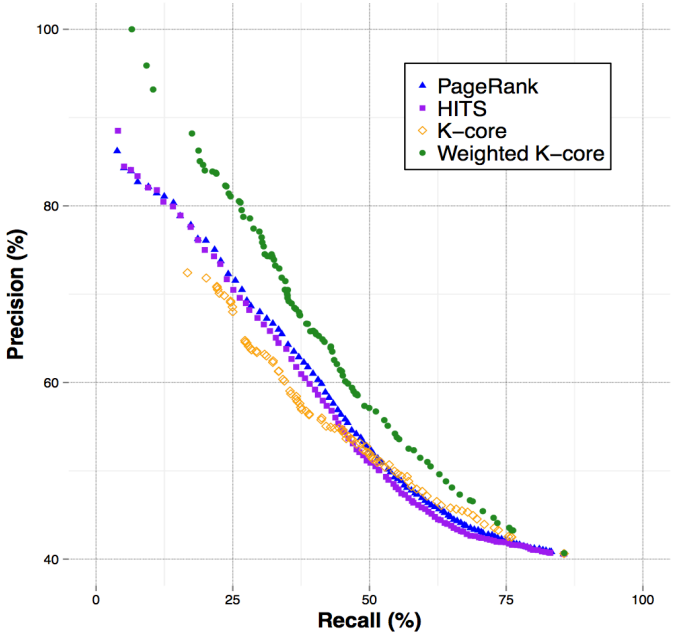
Performance Evaluation

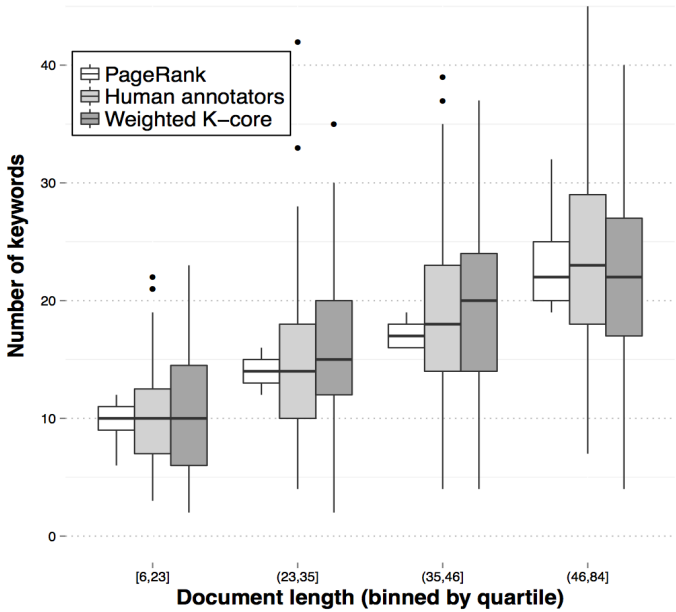
Precision

Recall

F1-score

Precision/recall





Graph	Dataset	Macro-averaged precision (%)				Macro-averaged recall (%)				Macro-averaged F1-score (%)			
		PageRank	HITS	K-core	WK-core	PageRank	HITS	K-core	WK-core	PageRank	HITS	K-core	WK-core
undirected edges	Hulth2003	58.94	57.86	46.52	**61.24***	42.19	41.80	**62.51***	50.32*	47.32	46.62	49.06*	**51.92***
	Krapi2009	50.23	49.47	40.46	**53.47***	48.78	47.85	**78.36***	50.21	49.59	47.96	46.61	**50.77***
forward edges	Hulth2003	55.80	54.75	42.45	**56.99***	41.98	40.43	**72.87***	46.93*	45.70	45.03	**51.65***	50.59*
	Krapi2009	47.78	47.03	39.82	**52.19***	44.91	44.19	**79.06***	45.67	45.72	44.95	46.03	**47.01***
backward edges	Hulth2003	59.27	56.41	40.89	**60.24***	42.67	40.66	**70.57***	49.91*	47.57	45.37	45.20	**50.03***
	Krapi2009	51.43	49.11	39.17	**52.14***	49.96	47.00	**77.60***	50.16	**50.51**	47.38	46.93	50.42

Example – ECIR'15 Paper

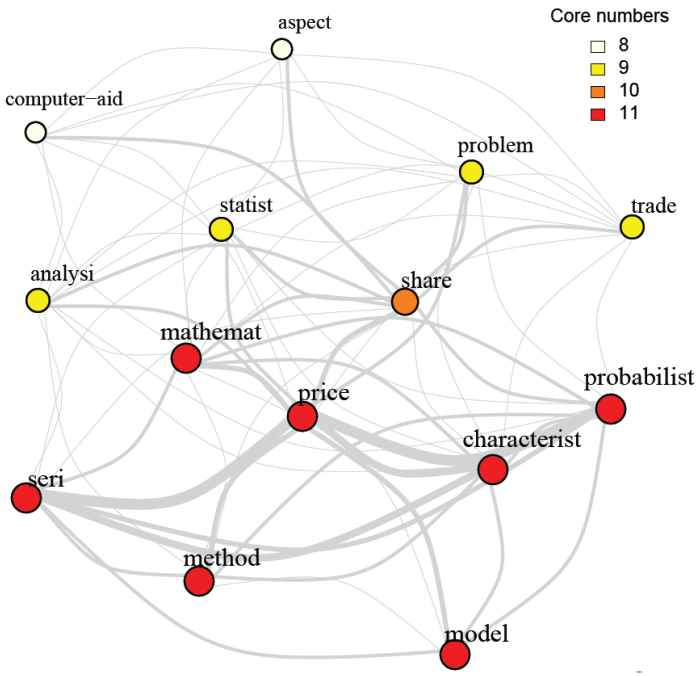
- Stemmed unigrams of the main core of the graph- of-words of the paper document: {*keyword, extract, graph, represent, text, weight, graph-of-word, k-core, degeneraci, edg, vertic, number, document*}

- Using PageRank, "work" appears in the top 5, "term" and "pagerank" in the top 10, and "case" and "order" in the top 15. Central words but not in cohesion with the rest and probably not relevant

A Different Point of View

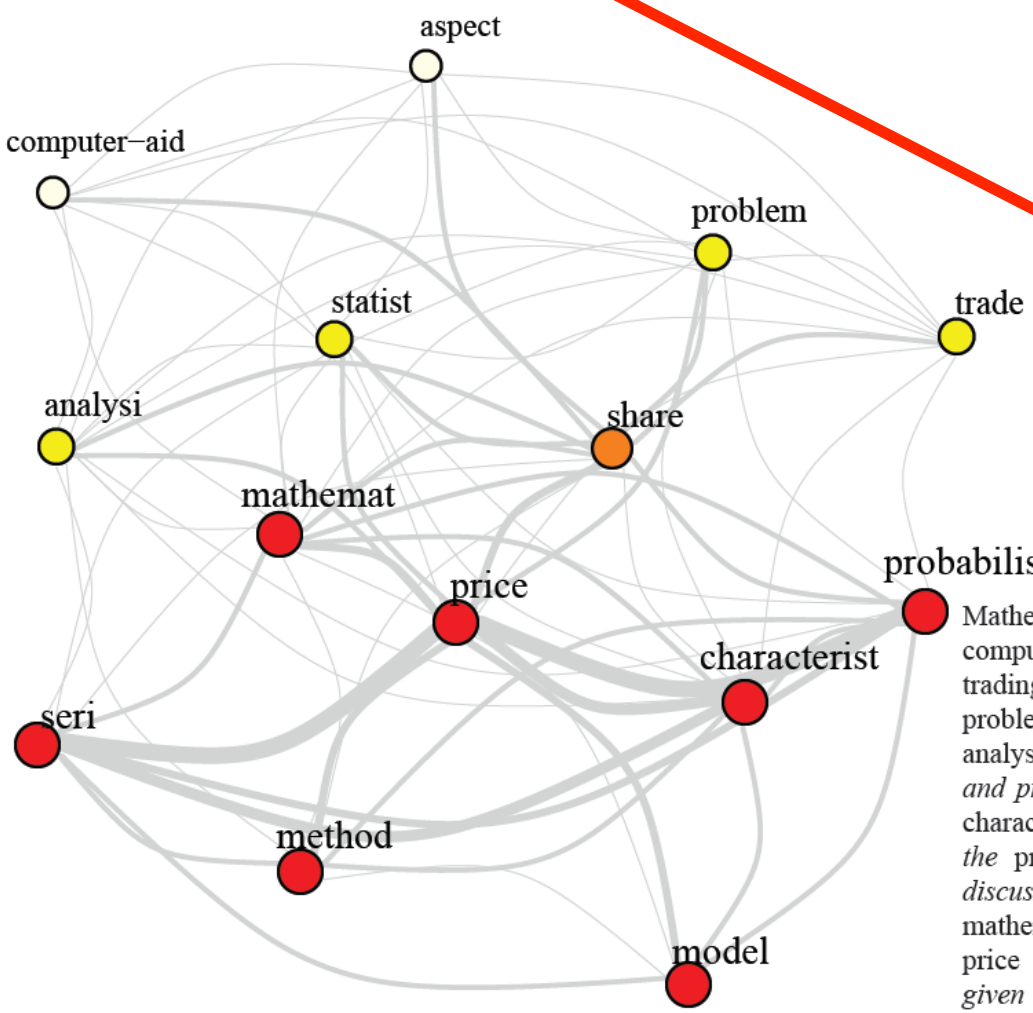
Graph degeneracy:

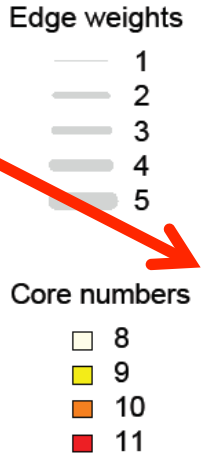
- In social networks, nodes part of the highest levels of the hierarchy are better spreaders than nodes high on PageRank
- Nodes with high truss numbers are even more influential than nodes with high core numbers
- Spreading influence may be a better "keywordness" metric than prestige (captured by PageRank)



Drawbacks of Graph Degeneracy (1/4)

Retaining the top level like in may be an appealing initial idea





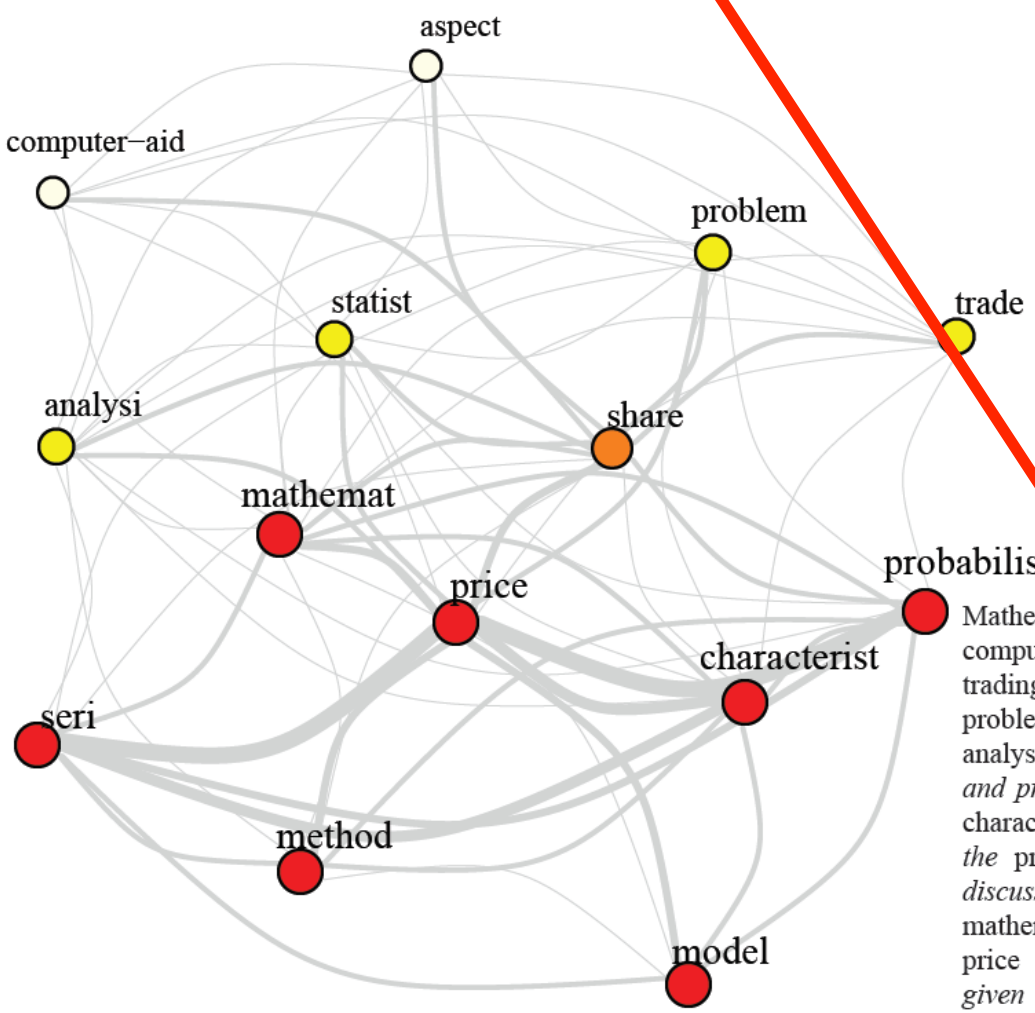
Mathematical aspects *of* computer-aided share trading. *We consider* problems *of* statistical analysis *of* share prices *and propose* probabilistic characteristics *to describe the* price series. *We discuss three* methods *of* mathematical modelling *of* price series *with given* probabilistic characteristics.

Core numbers			
	P	R	F1
MAIN	0.86	0.55	0.67
INF	**0.83**	**0.91**	**0.87**
DENS	0.88	0.64	0.74

mathemat		11
price		11
probabilist		11
characterist		11
seri		11
method		11
model	*MAIN*	11
share	*DENS*	10
trade		9
problem		9
statist		9
analysi	*INF*	9
aspect		8
computer-aid		8

Drawbacks of Graph Degeneracy (2/4)

But many keywords live below the top level -> good precision, poor recall



Edge weights
1
2
3
4
5

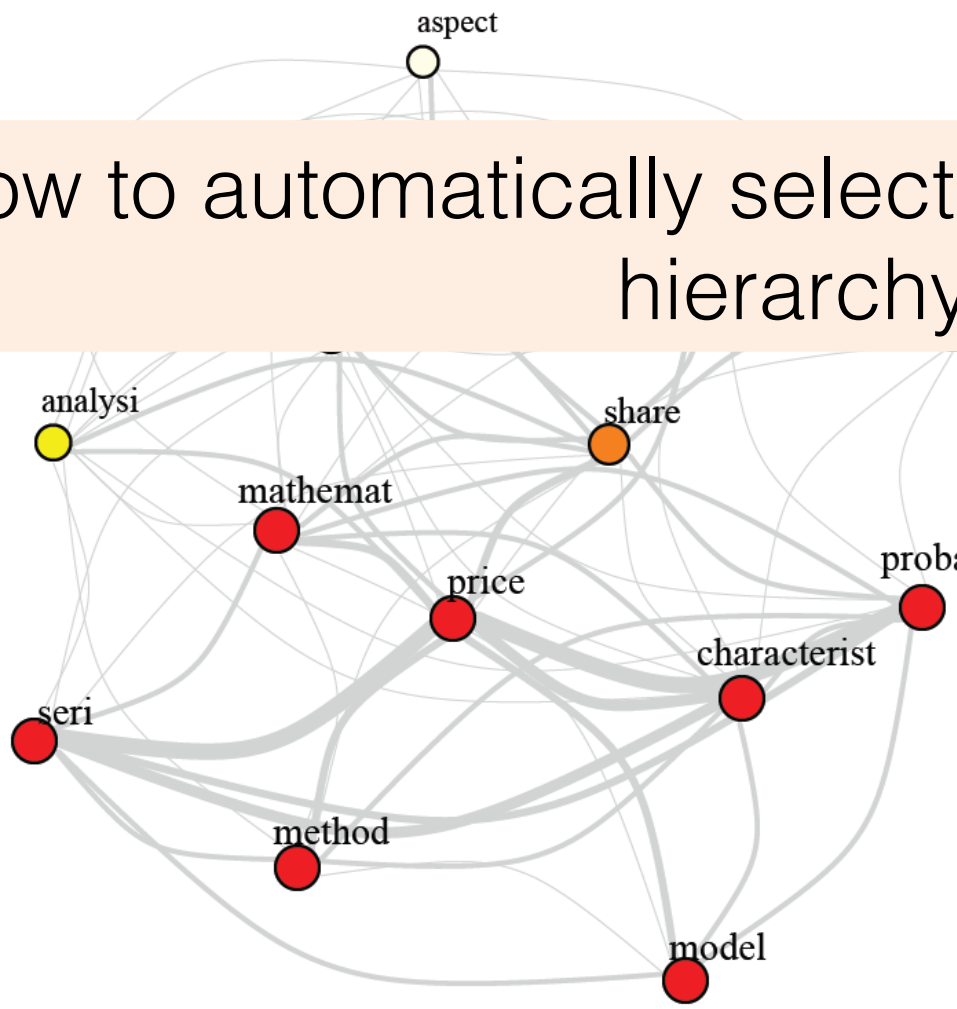
Core numbers
8
9
10
11

Mathematical aspects *of* computer-aided share trading. *We consider* problems *of* statistical analysis *of* share prices *and propose* probabilistic characteristics *to describe the* price series. *We discuss three* methods *of* mathematical modelling *of* price series *with given* probabilistic characteristics.

Core numbers			
	P	R	F1
MAIN	0.86	0.55	0.67
INF	**0.83**	**0.91**	**0.87**
DENS	0.88	0.64	0.74
mathemat			11
price			11
probabilist			11
characterist			11
seri			11
method			11
model *MAIN*			11
share *DENS*			10
trade			9
problem			9
statist			9
analysi *INF*			9
aspect			8
computer-aid			8

Drawbacks of Graph Degeneracy (3/4)

How to automatically select the best level in the hierarchy?



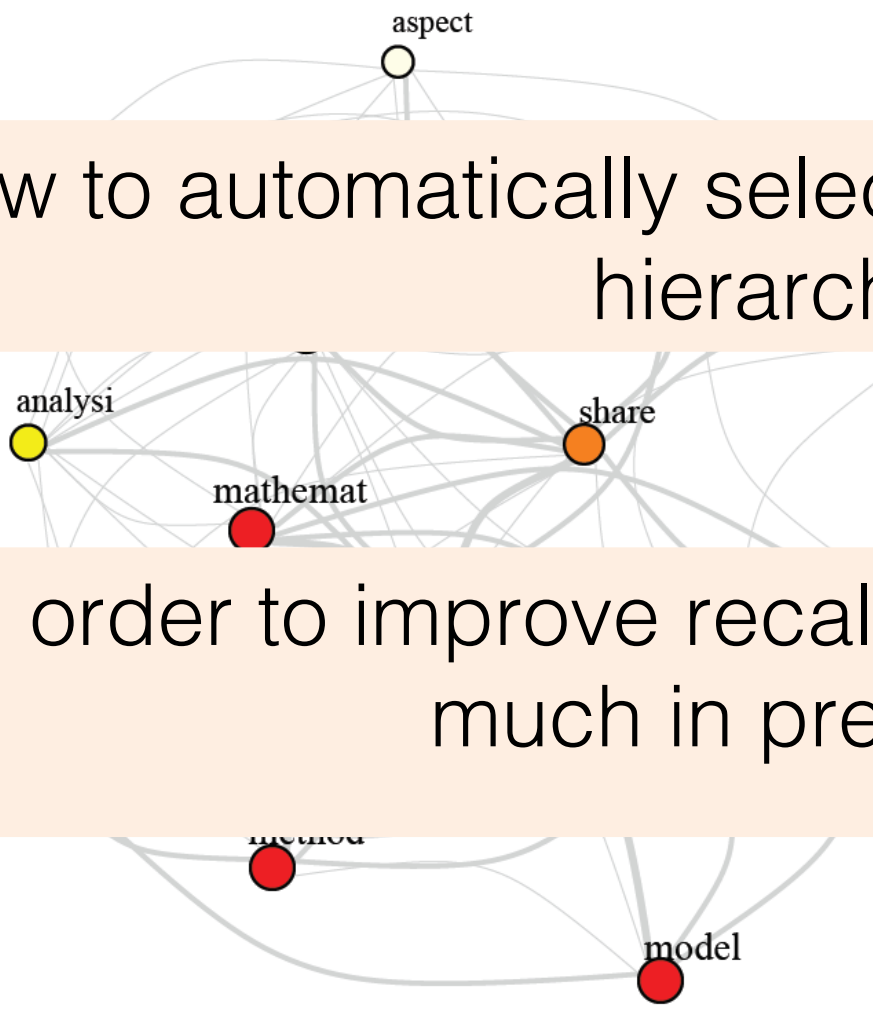
Edge weights
1

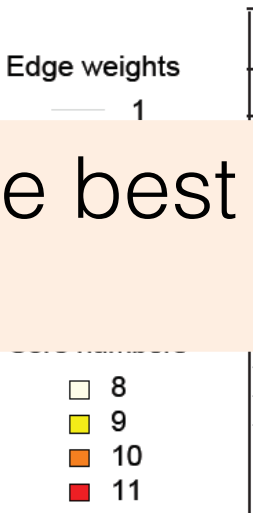
8
9
10
11

Mathematical aspects *of* computer-aided share trading. *We consider* problems *of* statistical analysis *of* share prices *and propose* probabilistic characteristics *to describe the* price series. *We discuss three* methods *of* mathematical modelling *of* price series *with given* probabilistic characteristics.

Core numbers			
	P	R	F1
probabilist			11
characterist			11
seri			11
method			11
model	*MAIN*		11
share	*DENS*		10
trade			9
problem			9
statist			9
analysi	*INF*		9
aspect			8
computer-aid			8

Drawbacks of Graph Degeneracy (4/4)





	P	R	F1
price			11
probabilist			11
characterist			11
seri			11

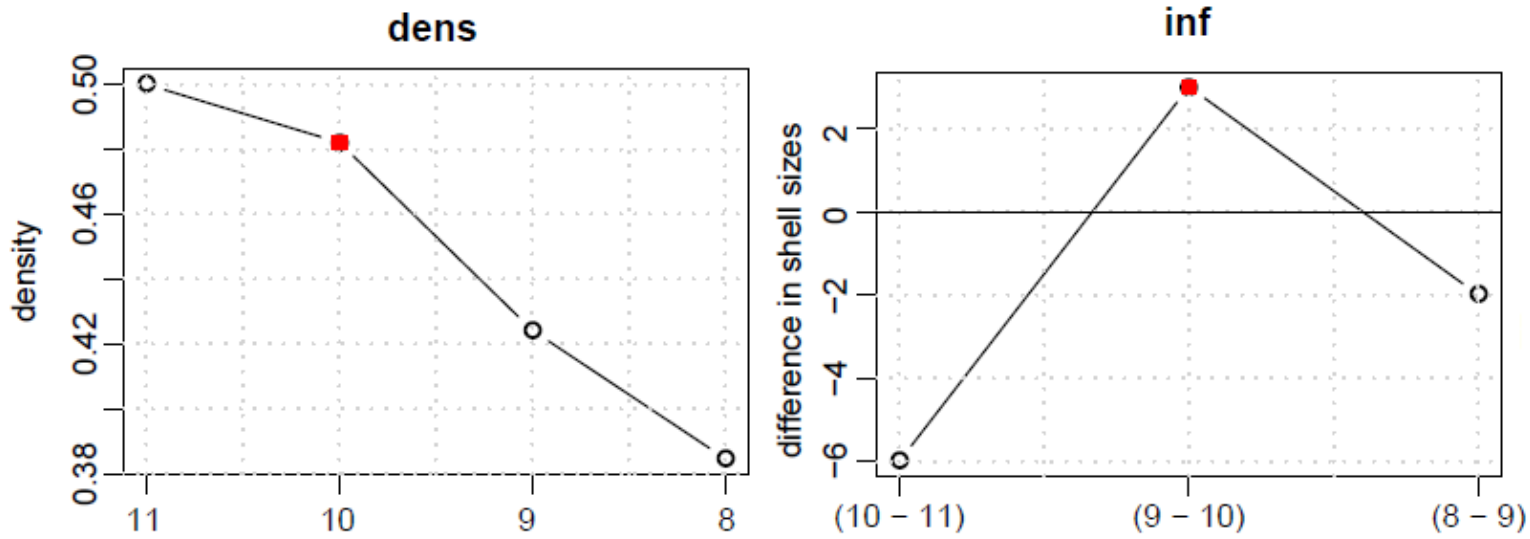
How to automatically select the best level in the hierarchy?

In order to improve recall while not losing too much in precision?

the price series. *We discuss three* methods *of* mathematical modelling *of* price series *with given* probabilistic characteristics.

statist		9
analysi	*INF*	9
aspect		8
computer-aid		8

Graph Degeneracy for Keyword Extraction



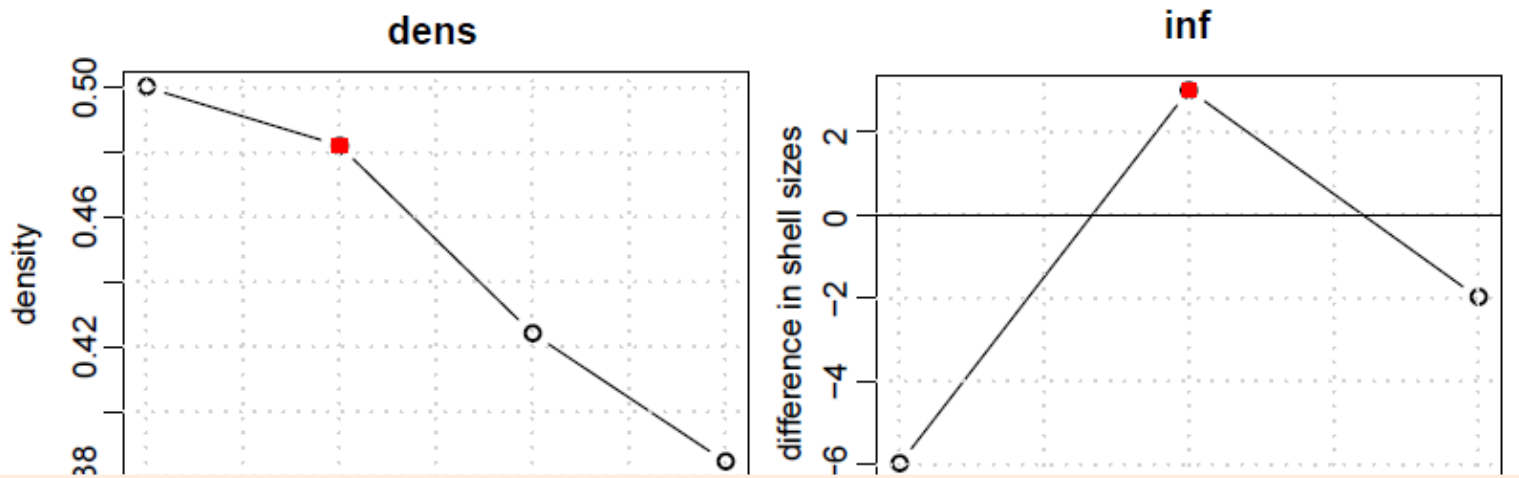
Heuristics:

- dens: go down the hierarchy until a drop in k-core (or truss) density is observed, i.e., as long as the desirable cohesiveness properties are kept
- inf: go down the hierarchy as long as the shells increase in size (starting at the main – 1 level)

Problem: both methods work at the subgraph level -> lack flexibility for large graphs (adding an entire group of nodes or not)

[Tixier et al., EMNLP '16]

Graph Degeneracy for Keyword Extraction



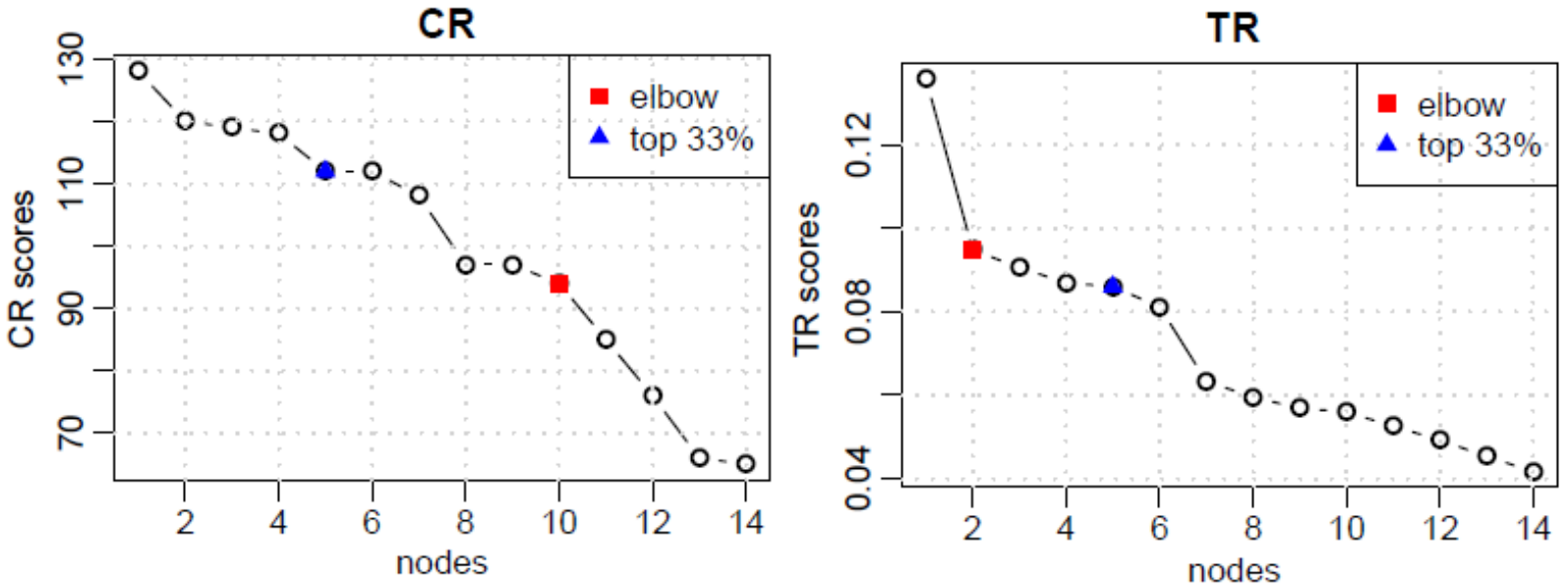
How to work at the node level while still retaining the valuable cohesiveness information captured by degeneracy?

- inf: go down the hierarchy as long as the shells increase in size (starting at the main – 1 level)

Problem: both methods work at the subgraph level -> lack flexibility for large graphs (adding an entire group of nodes or not)

[Tixier et al., EMNLP '16]

CoreRank



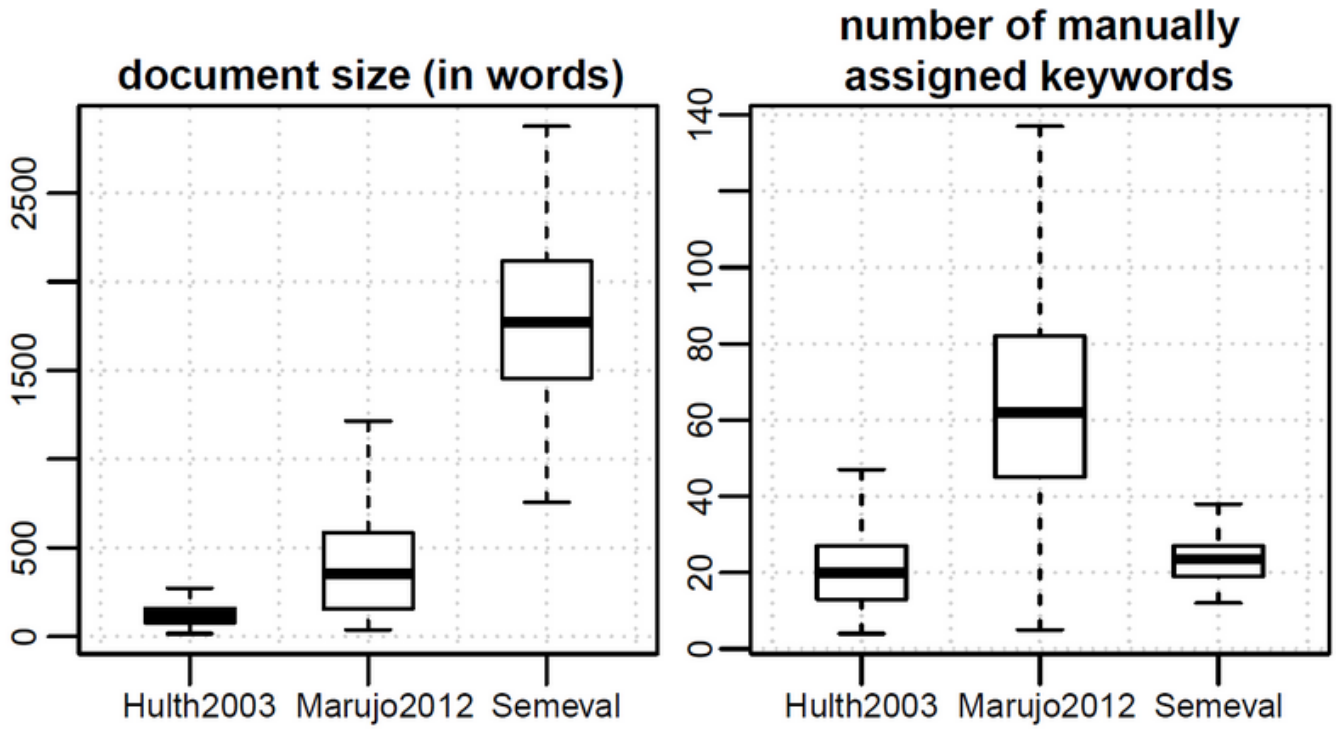
CoreRank (CR):

- Assign to each node the sum of the core (or truss) numbers of its neighbors
- Granularity is much finer and allows for much flexible selection
- Comparable to applying PageRank to the graph-of-words (aka TextRank) but taking into account cohesiveness concerns rather than individual prestige only

Heuristics: nodes can be selected based on the elbow or top p% method

[Tixier et al., EMNLP '16]

CoreRank – Experimental Evaluation (1/2)



Datasets

- Hulth2003: 500 abstracts from the Inspec physics & engineering database
- Marujo2012: 450 web news stories covering 10 different topics
- Semeval: 100 scientific papers from the ACM

CoreRank – Experimental Evaluation (2/2)

	precision	recall	F1-score
dens	**48.79**	**72.78**	**56.09***
inf	48.96	72.19	55.98*
CRP	61.53	38.73	45.75
CRE	65.33	37.90	44.11
main†	51.95	54.99	50.49
TRP†	65.43	41.37	48.79
TRE†	71.34	36.44	45.77

Hulth2003, K-truss, $W = 11$. *stat. sign. ($p < 0.001$) w.r.t. all baselines†

	precision	recall	F1-score
dens	**47.62**	**71.46**	**52.94***
inf	53.88	57.54	49.10*
CRP	54.88	36.01	40.75
CRE	63.17	25.77	34.41
main†	64.05	34.02	36.44
TRP†	55.96	36.48	41.44
TRE†	65.50	21.32	30.68

Marujo2012, k-core, $W = 13$. *stat. sign. ($p < 0.001$) w.r.t. all baselines†

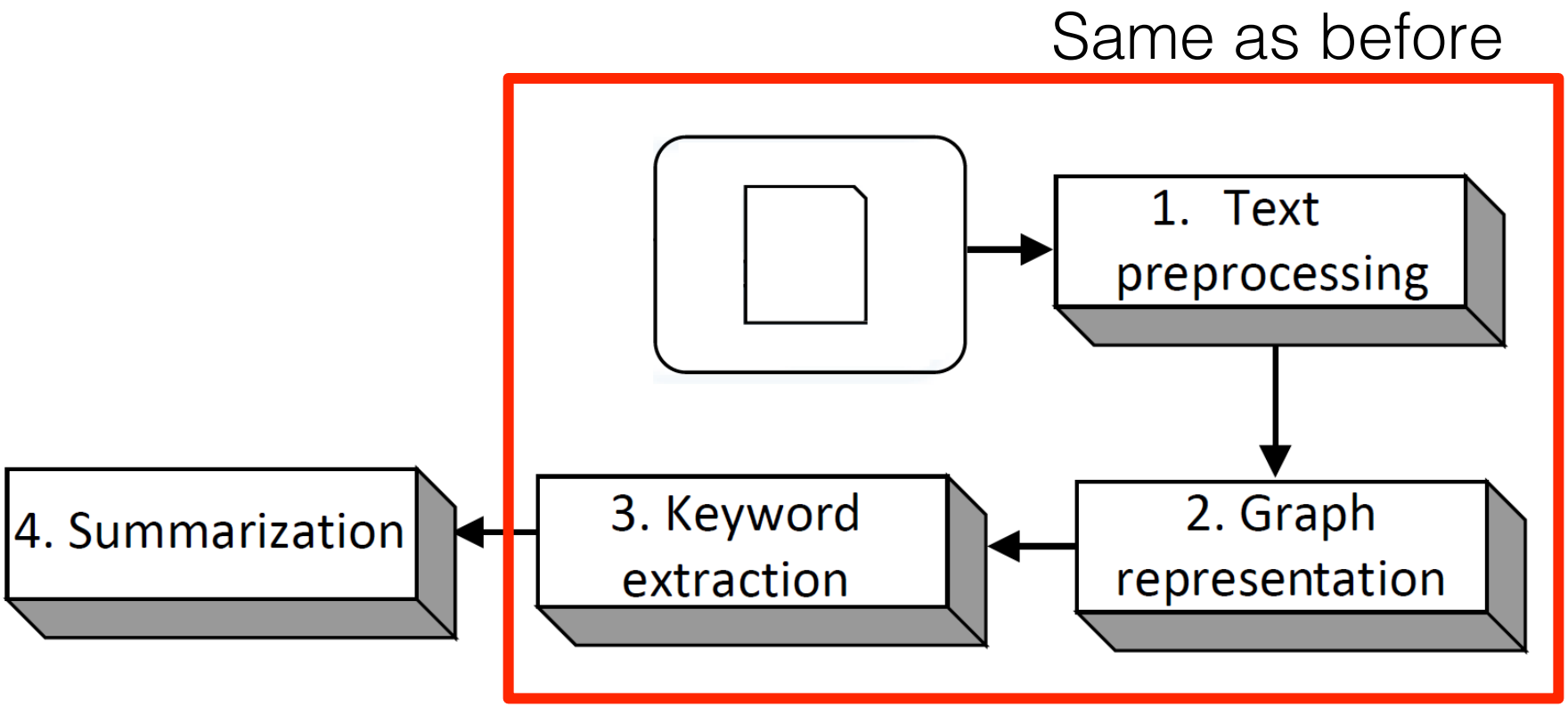
	precision	recall	F1-score
dens	8.44	79.45	15.06
inf	17.70	65.53	26.68
CRP	**49.67**	**32.88**	**38.98***
CRE	25.82	58.80	34.86
main†	25.73	49.61	32.83
TRP†	47.93	31.74	37.64
TRE†	33.87	46.08	37.55

Semeval, K-truss, $W = 20$. *stat. sign. ($p < 0.001$) w.r.t. *main*

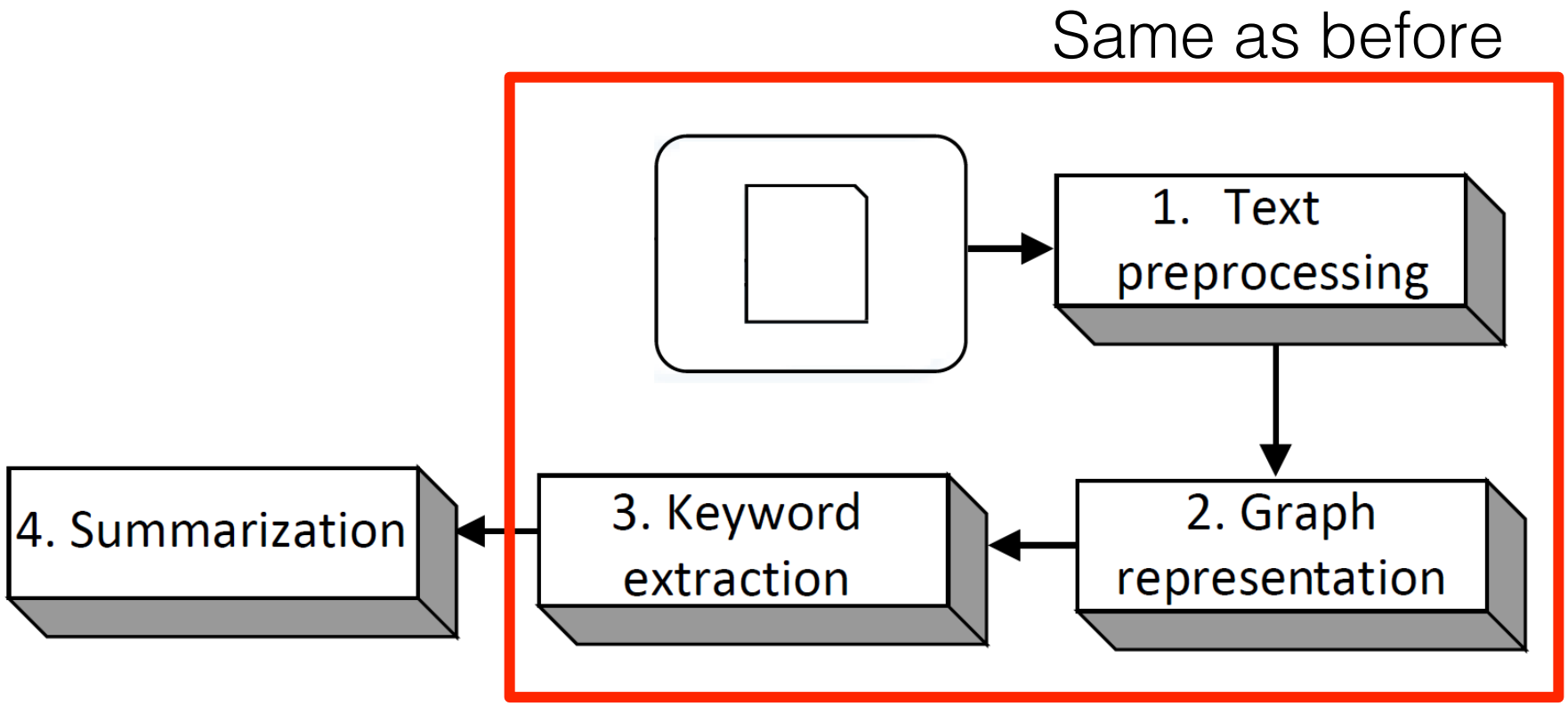
- For small documents (i.e., small graphs), the subgraph-level heuristics significantly outperform main core retention (main) and TextRank (TRP, TRE)
- Recall is drastically improved, precision is maintained (especially with inf)
- For long documents (Semeval), the node-level heuristics are better
- CoreRank with top p% retention (CRP) reaches best performance

Extractive summarization

Extension to Extractive Document Summarization



Extension to Extractive Document Summarization



How to use keywords (and their scores) to select the best sentences in a document?

Extractive Document Summarization (1/4)

- Generating a summary in an extractive way is akin to selecting the best sentences in the document under a budget constraint (max number of words allowed)
- Combinatorial optimization task:

$$\arg\max_{S \subseteq V} F(S) \mid \sum_{v \in S} c_v \leq B$$

  - S is a given summary (a subset of the set of sentences V)
  - F is the objective function to maximize (measuring summary quality)
  - C_v is the cost of sentence v (number of words it contains)
  - B is the budget (in words)

[Lin and Bilmes, NAACL '10]

Extractive Document Summarization (2/4)

$$\arg\max_{S \subseteq V} F(S) \mid \sum_{v \in S} c_v \leq B$$

- Solving this task is NP-complete
- It has been shown that if F is non-decreasing and submodular, a greedy
  algorithm can approach the best solution with factor (e – 1)/e
- At each step, the algorithm selects the sentence v that maximizes:

objective function gain →

$$\frac{F(G \cup v) - F(G)}{c_v^r}$$

scaled cost →

- r is a tuning parameter

Extractive Document Summarization (3/4)

$$\arg\max_{S \subseteq V} F(S) \mid \sum_{v \in S} c_v \leq B$$

- The choice of F, the summary quality objective function, is what matters
- A good summary should cover all the important topics in the document, while not repeating itself
  - Maximize coverage
  - Penalize redundancy (reward diversity to ensure monotonicity)

$$F(S) = L(S) + \lambda R(S)$$

$$L(S) = \sum_{i \in S} n_i w_i \qquad R(S) = N_{keywords \in S} / N_{keywords}$$

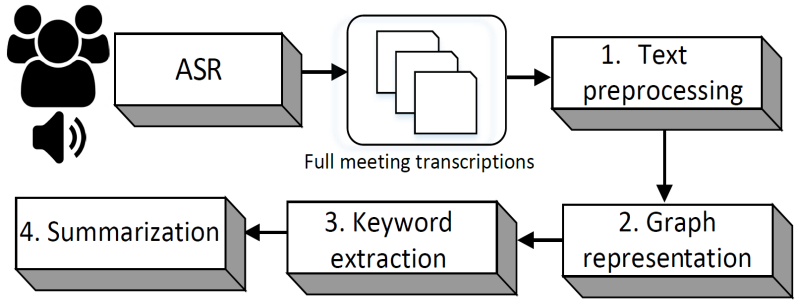
weighted sum of the keywords contained in the summary

proportion of unique keywords contained

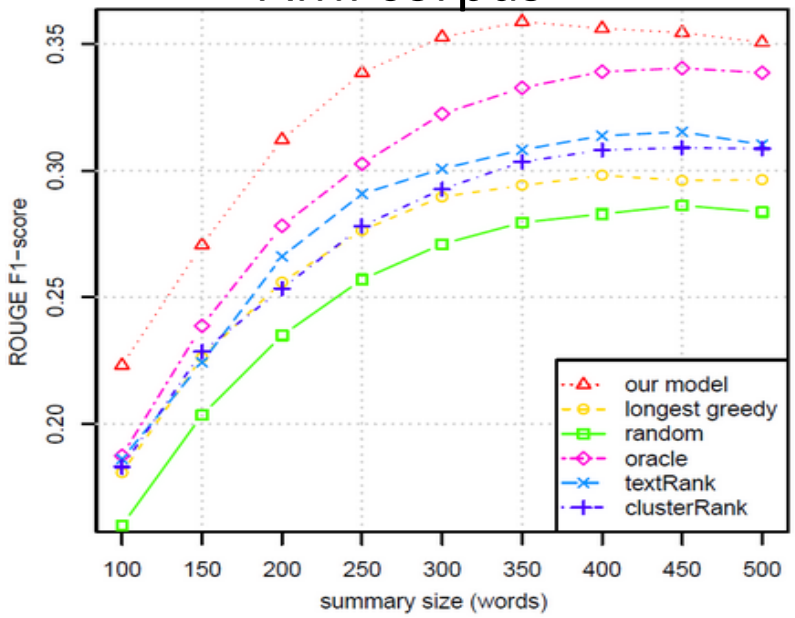
[Lin and Bilmes, NAACL '10], [Meladianos et al., EACL '17]

Extractive Document Summarization (4/4)

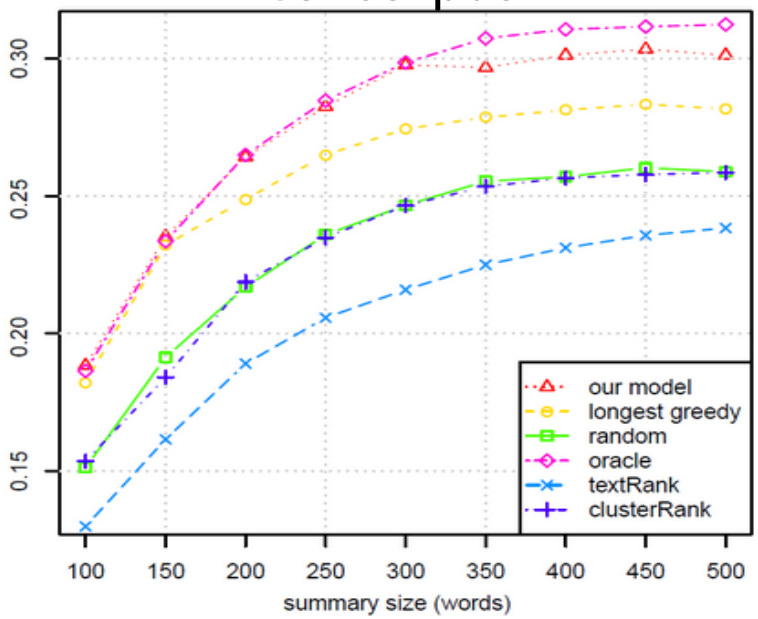
Tested for multiparty virtual meetings summarization:



AMI corpus

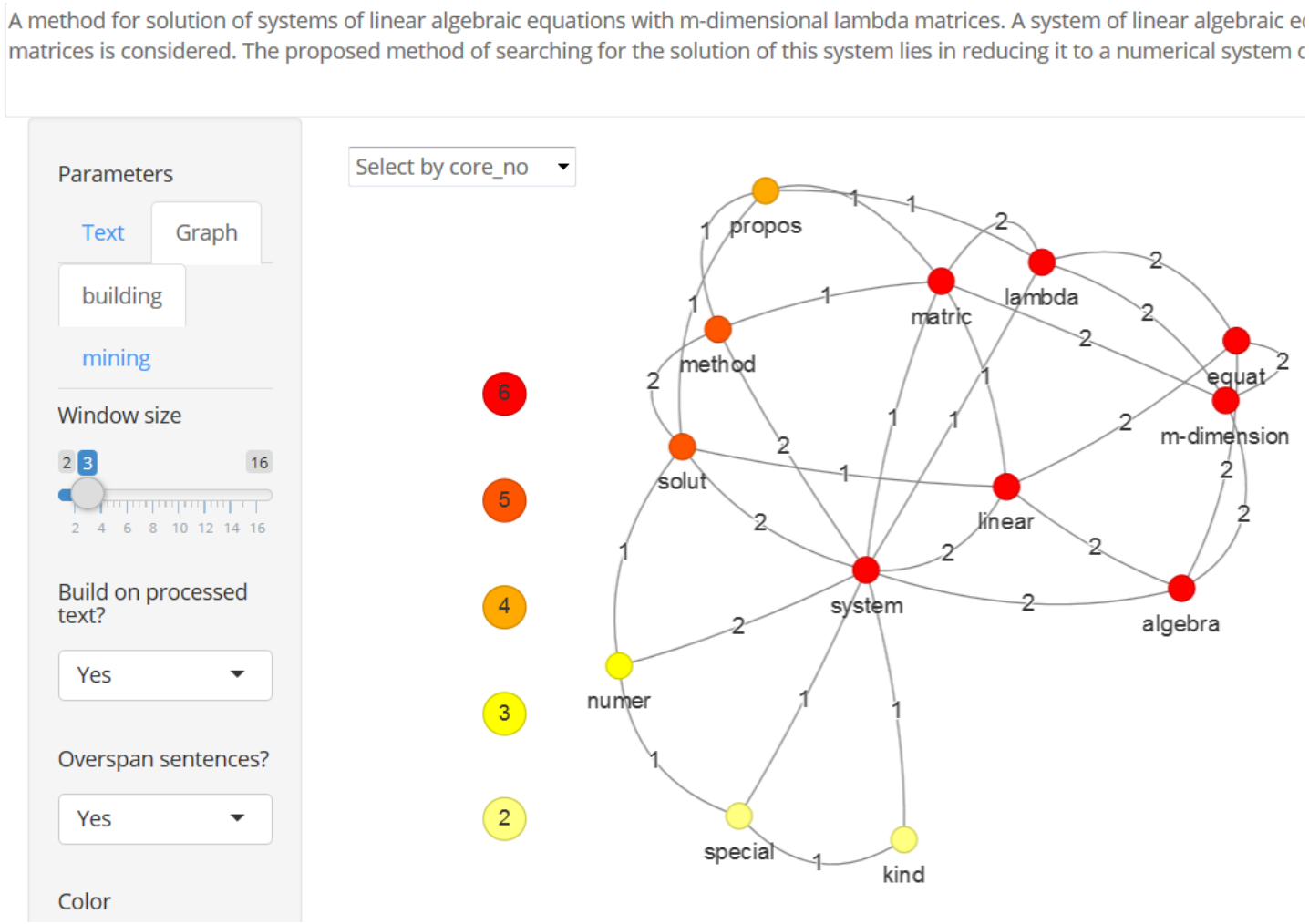


ICSI corpus



GoWvis visualization tool

GoWvis Visualization Tool



https://safetyapp.shinyapps.io/GoWvis/

[Tixier et al., ACL '16]

GoWvis

- Builds a graph-of-words and displays an interactive representation of any text pasted by the user
- Allows the user to tune many parameters:
  - Text pre-processing (stopword removal, …)
  - Graph building (window size, …)
  - Graph mining (node ranking and community detection algorithms, …)
- Extracts keyphrases and generates a summary of the input text
- Built in R Shiny with the visNetwork library

https://safetyapp.shinyapps.io/GoWvis/

Multi-sentence compression in word graphs

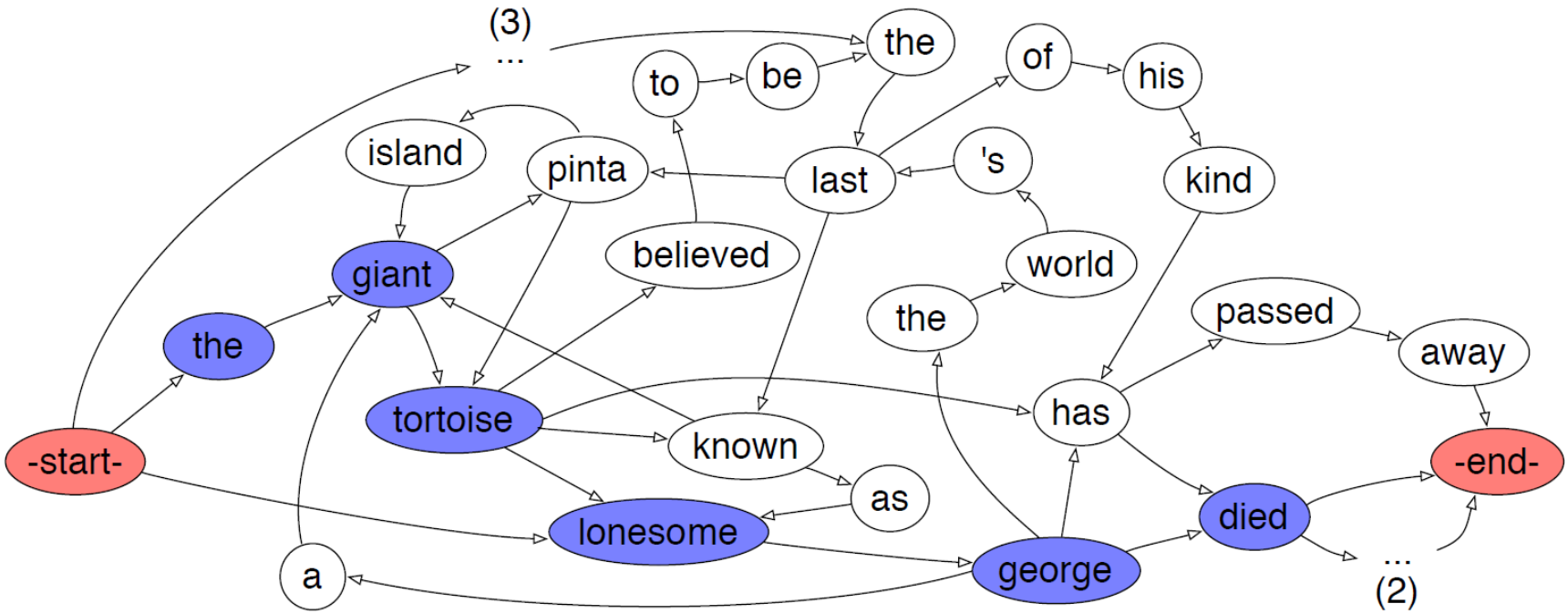
Multi Sentence Compression/Fusion

- Setting: we are given a group of similar sentences (e.g., first sentence of each article on a Google News cluster). Each sentence contains important bits of information. Collectively, the sentences cover everything, but none single sentence 'gets it all'
- Goal: fuse the sentences into a single, compact one, that contains as much information as possible while being fluent and grammatical
- Method: many approaches can be used. However, it is possible to produce excellent results in a fully unsupervised way, with only a list of stopwords and a part-of-speech tagger

[Filippova et al., ACL '10]

Word-graph Sentence Compression

1) Lonesome George, the world's last Pinta Island giant tortoise, has passed away
2) The giant tortoise known as Lonesome George died Sunday at the Galapagos National Park in Ecuador
3) He was only about a hundred years old, but the last known giant Pinta tortoise, Lonesome George, has passed away
4) Lonesome George, a giant tortoise believed to be the last of his kind, has died



Word-graph Construction

- Build a directed graph from the first sentence, with 'start' and 'end' nodes. Then, consider each word in the remaining sentences.

i. if the word is not a stopword, and if there is already a node in the graph for it (with same lowercased spelling and POS tag), and assuming that no word from the same sentence has already been mapped onto the node => map word to the node

- Otherwise:

ii. if the word is not a stopword, but there are more than one candidate in the graph or multiple occurrences of the word in the sentence

iii. if the word is a stopword

=> select the candidate which has larger overlap in context (preceding and following words in sentence and neighbors in the graph), or the node which has more words mapped onto it

Word-graph Construction

Edge weights (the smaller the better):

$$w''(e_{i,j}) = \frac{w'(e_{i,j})}{\texttt{freq}(i) \times \texttt{freq}(j)} \quad (1)$$

Where:

$$w'(e_{i,j}) = \frac{\texttt{freq}(i) + \texttt{freq}(j)}{\sum_{s \in S} \texttt{diff}(s,i,j)^{-1}} \quad (2)$$

- freq(i) is the number of words that have been mapped to node i
- diff(s,i,j) is the distance between word i and word j in sentence s

- Intuition for (2):
  - edges between strongly associated words are given more importance, taking into account the overall freq. of the nodes (edge freq. of 3 should count more if the edge connects 2 nodes with freq. 3 rather than with freq. >>3)
  - Connections between nodes between which there are multiple paths are also given more importance, proportionally to the lengths of the paths

Word-graph Construction

Edge weights (the smaller the better):

$$w''(e_{i,j}) = \frac{w'(e_{i,j})}{\texttt{freq}(i) \times \texttt{freq}(j)} \quad (1)$$

Where:

$$w'(e_{i,j}) = \frac{\texttt{freq}(i) + \texttt{freq}(j)}{\sum_{s \in S} \texttt{diff}(s,i,j)^{-1}} \quad (2)$$

- freq(i) is the number of words that have been mapped to node i
- diff(s,i,j) is the distance between word i and word j in sentence s

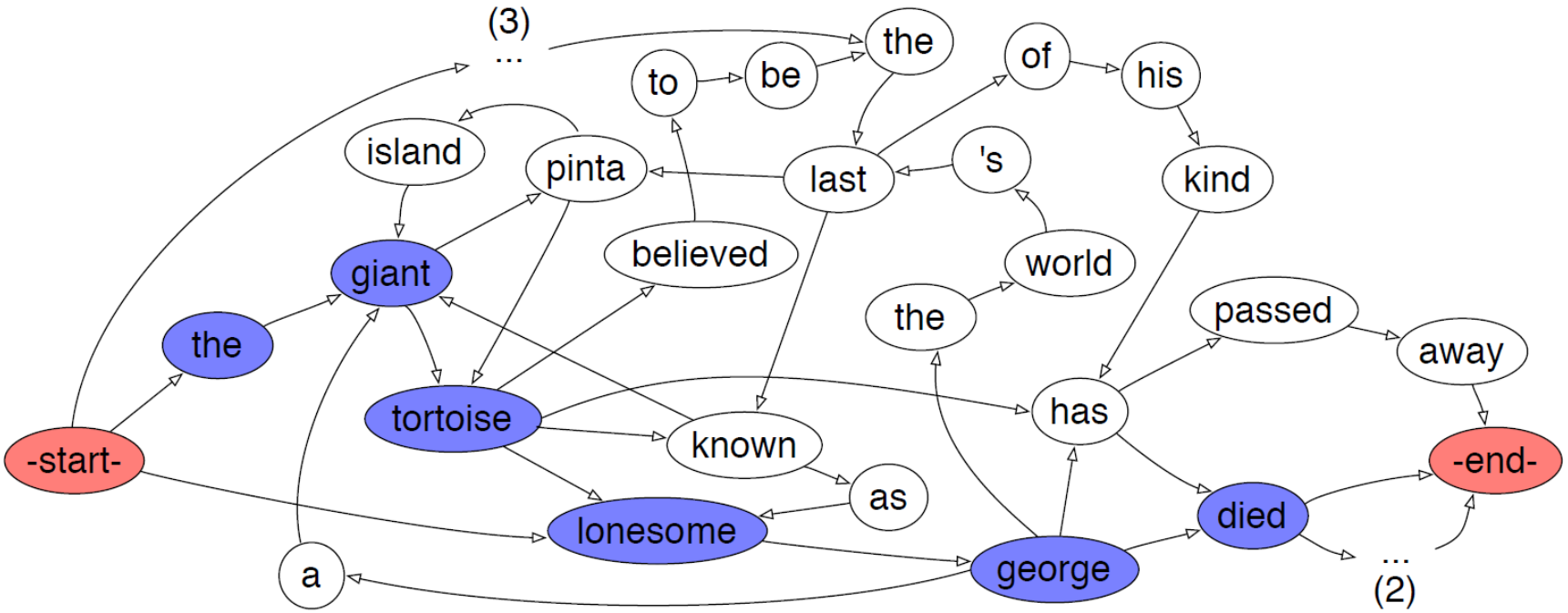
- Intuition for (1):
  - Eq. (2) is a measure of cohesion between 2 words, but disregards the individual importance of the words => we need to take saliency into account. Edges connecting two important words are thus favored.

Path Ranking and Selection

- A K-shortest paths algorithm is applied on the graph to find the 50 paths with smallest edge weights
- All the paths which are shorter than eight words and do not contain a verb are filtered out
- The survivors are re-ranked by normalizing the total path weight over its length
- The path which has the lightest average edge weight is finally considered as the best compression

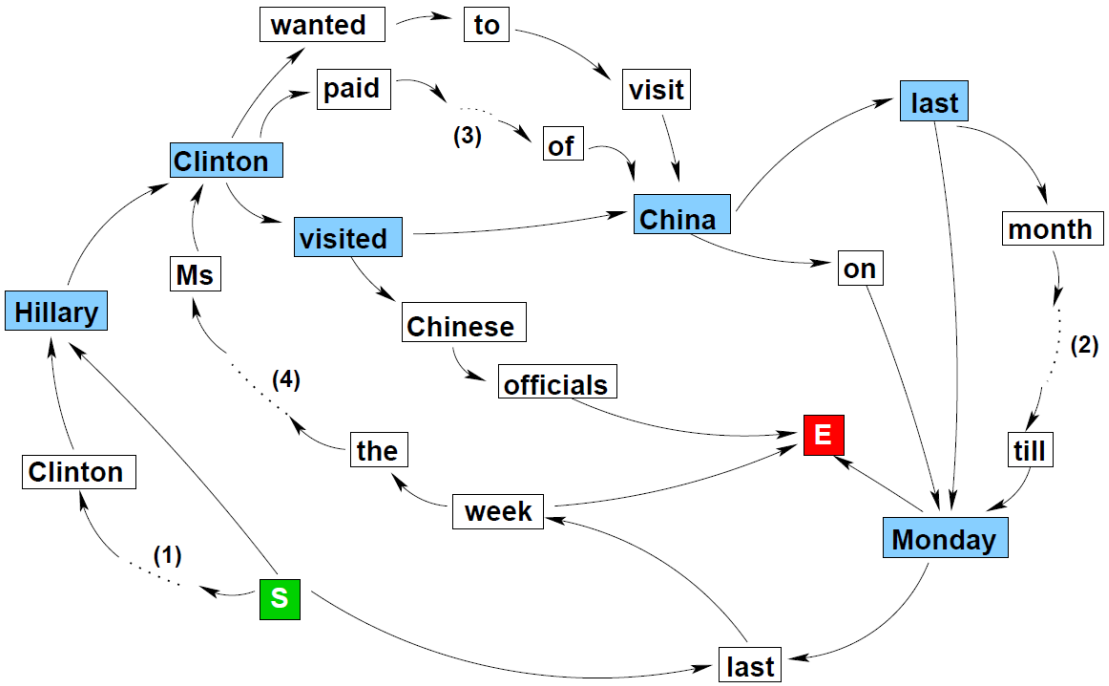
Examples

1) Lonesome George, the world's last Pinta Island giant tortoise, has passed away
2) The giant tortoise known as Lonesome George died Sunday at the Galapagos National Park in Ecuador
3) He was only about a hundred years old, but the last known giant Pinta tortoise, Lonesome George, has passed away
4) Lonesome George, a giant tortoise believed to be the last of his kind, has died



Examples

1) The wife of a former U.S. president Bill Clinton Hillary Clinton visited China last Monday
2) Hillary Clinton wanted to visit China last month but postponed her plans till Monday last week
3) Hillary Clinton paid a visit to the People Republic of China on Monday
4) Last week the Secretary of State Ms. Clinton visited Chinese officials



Lessons Learned

- Syntactic parsers, language models, and/or handcrafted rules are not the only way of controlling the grammaticality of the output
- Redundancy provides a reliable way of generating grammatical sentences

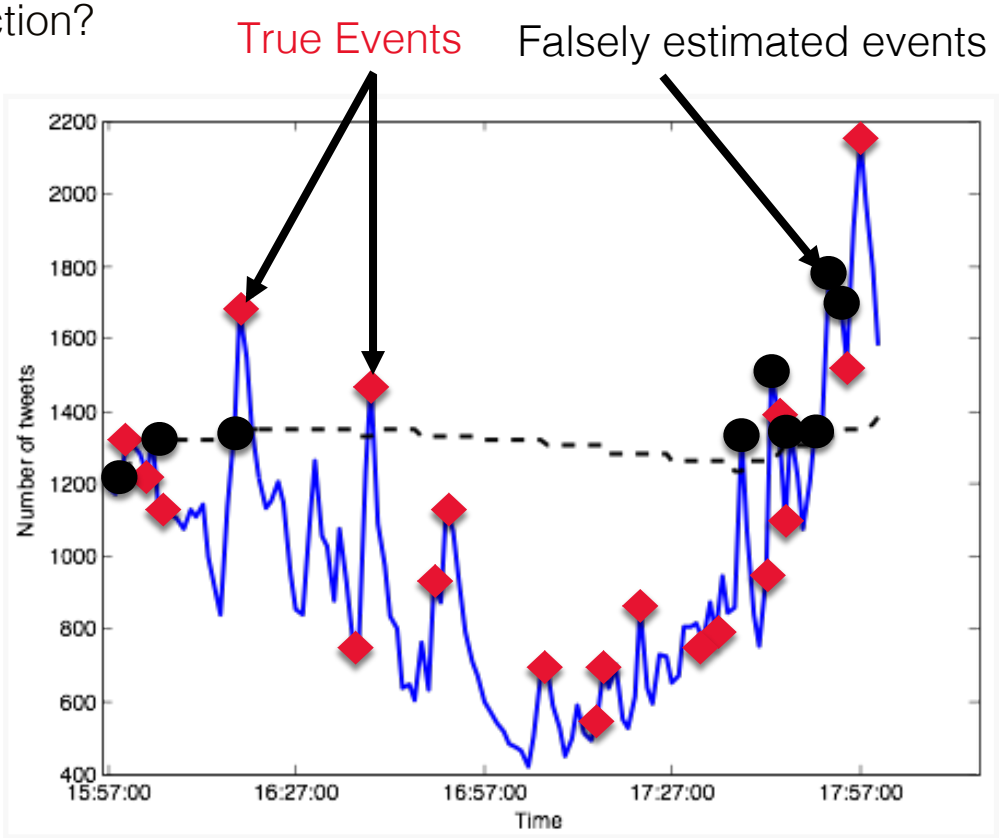
Event detection in text streams

Sub-event Detection in Twitter Streams

1. Large volume of documents in social media
2. Events are not covered by traditional media
3. News appear fast in Twitter
4. Is Tweet rate suited for sub-event Detection?

Contribution:

A novel real-time detection mechanism that accurately detects sub-events in an unsupervised and out-of-the-box manner



Tweet Rate histogram of a football match

[Meladianos et al., ICWSM '15]

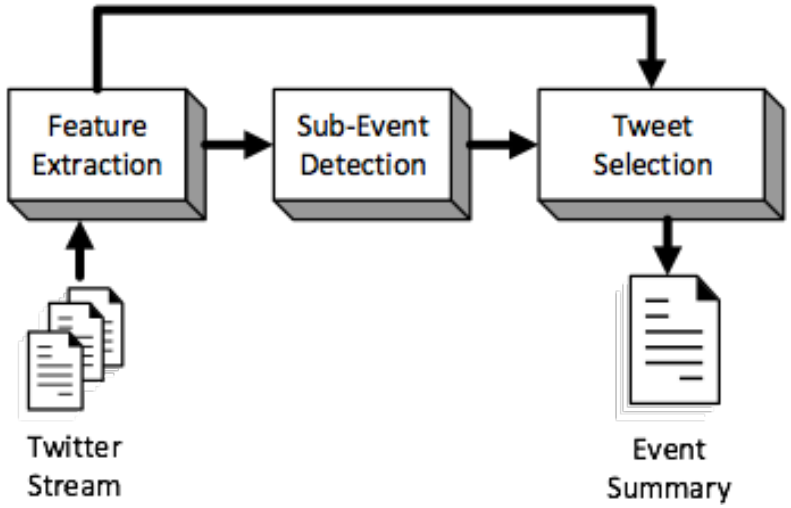
Sub-event Detection in Twitter Streams

Real time event summarization

1. Feature extraction: extracts the terms that best describe the current state of the event

2. Sub-event detection: decides whether a sub-event has occurred

3. Tweet selection: ranks all the tweets and selects the first one



System Architecture

- Steps are repeated every 60 seconds
- The summary of the whole event is constructed by aggregating the individual sub-event descriptions

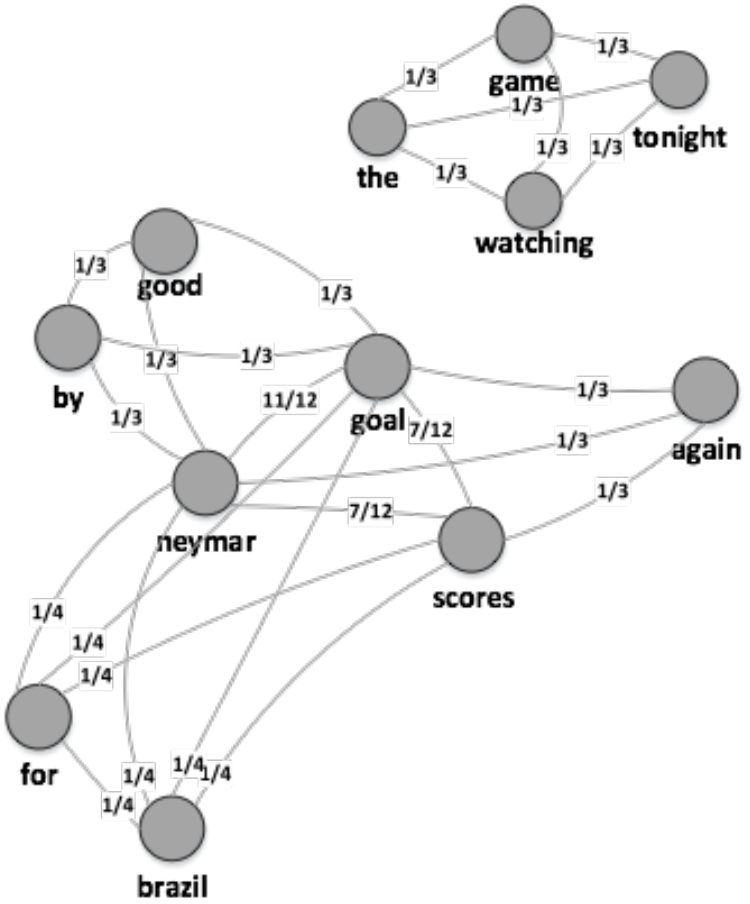
Graph-based Representation of Tweets

- Represents all the input tweets
- Nodes: unique terms
- Edges: #co-occurrences within a tweet

Example graph

1. Good goal by Neymar
2. Goal! Neymar scores for brazil
3. Goal!! Neymar scores again
4. Watching the game tonight

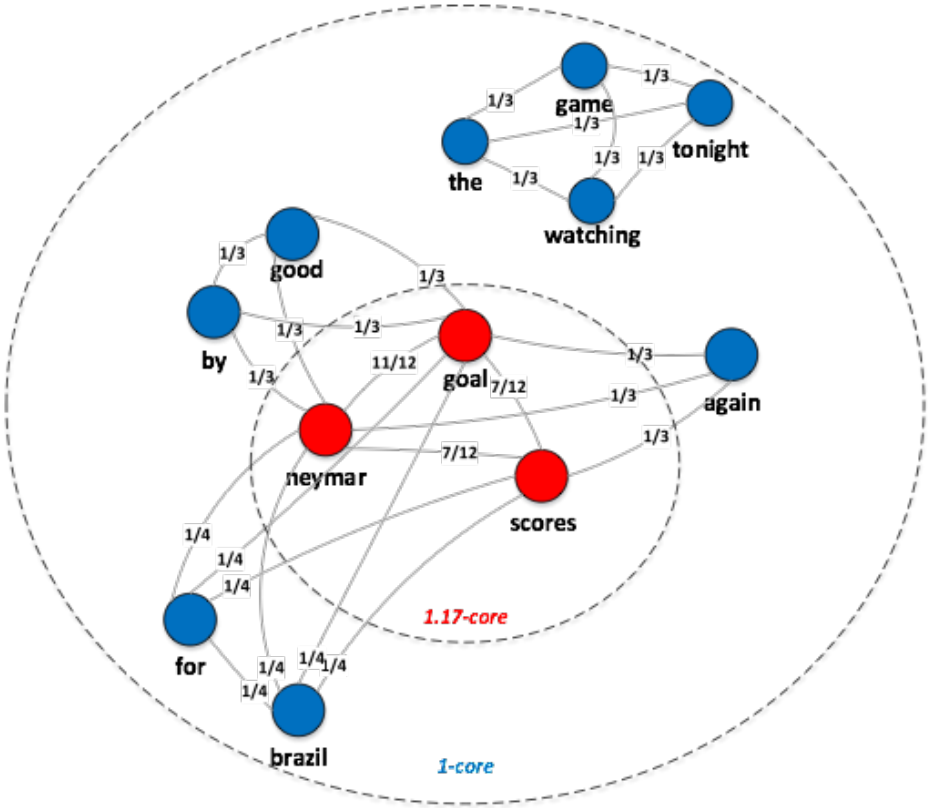
Dataset: tweets from the 2014 FIFA World Cup in Brazil



The graph that was built from 4 tweets

k-core Decomposition for Feature Extraction

- Each term is given a score corresponding to its core number
- Extract the k-core subgraph
- Detect sub-events by considering how the sum of the core numbers extracted from the graph at time t has changed from a previous time point t-1



k-core decomposition of the Graph-of-Words

Sub-event Detection

$$\sum_{i=1}^{d} c_i^t > \theta \times \frac{1}{p} \sum_{j=t-p}^{t-1} \sum_{i=1}^{d} c_i^j$$

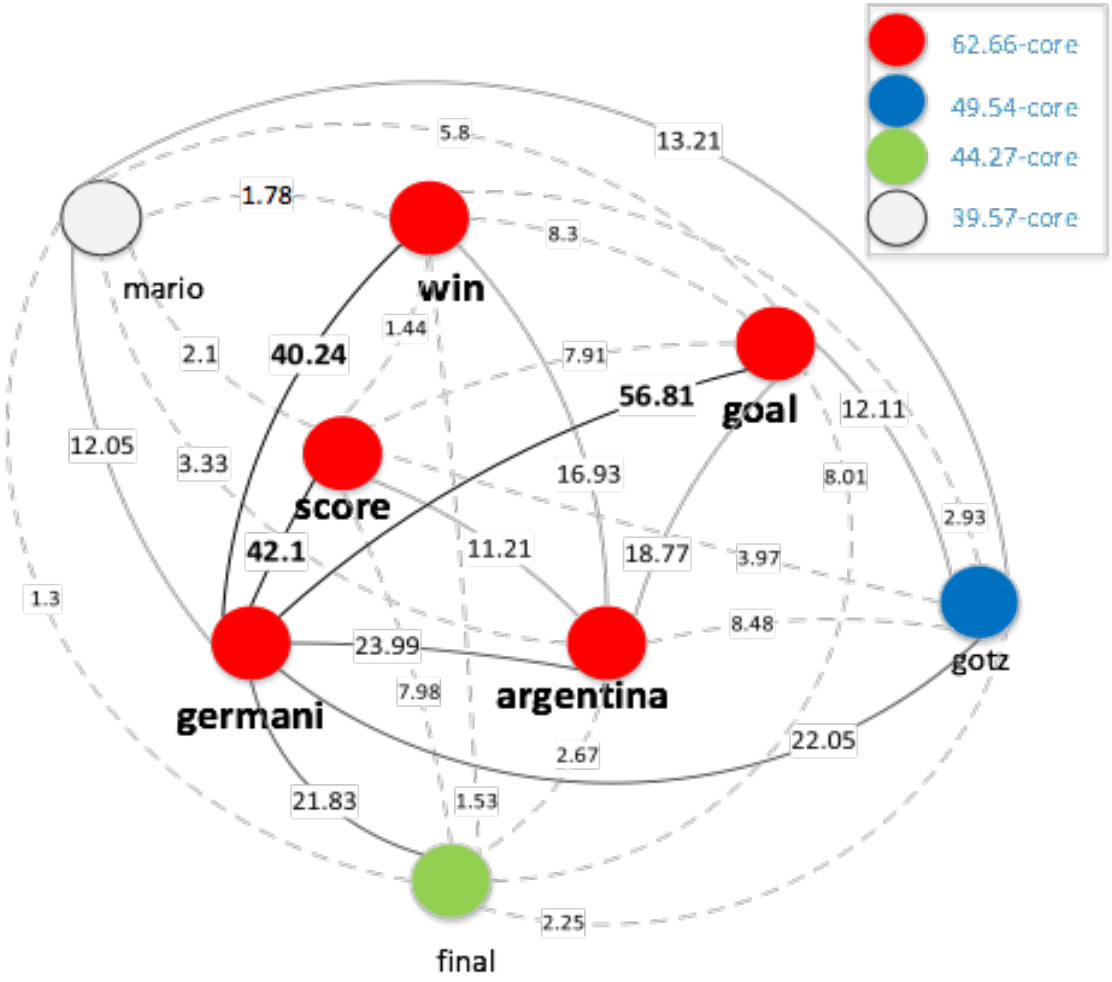
c_i^t	Core number of term at time slot
d	Number of terms selected
θ	Decision Threshold
p	Number of previous time slots

Sub-event Detection steps:

(every 60 seconds)

1. Extract the top terms with highest weights
2. Sum the term weights
3. If it exceeds the threshold a sub-event is detected

Germany's Goal - 2014 World Cup

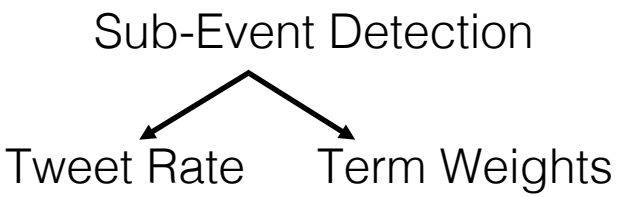


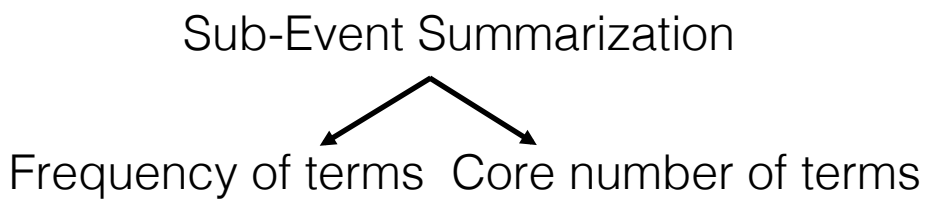
Snapshot of the four highest cores of the graph generated after Germany's goal in the 2014 FIFA World Cup final

Tweet Selection as Sub-event Summarization

- Activated only if a sub-event has been detected
- Tweets are scored based on the sum of their term weights
- Selects the most informative tweet of the sub-event
  - The tweet with the highest score is chosen

Expreimental Setup





Baselines-Approaches

(Detection -Term Weight)

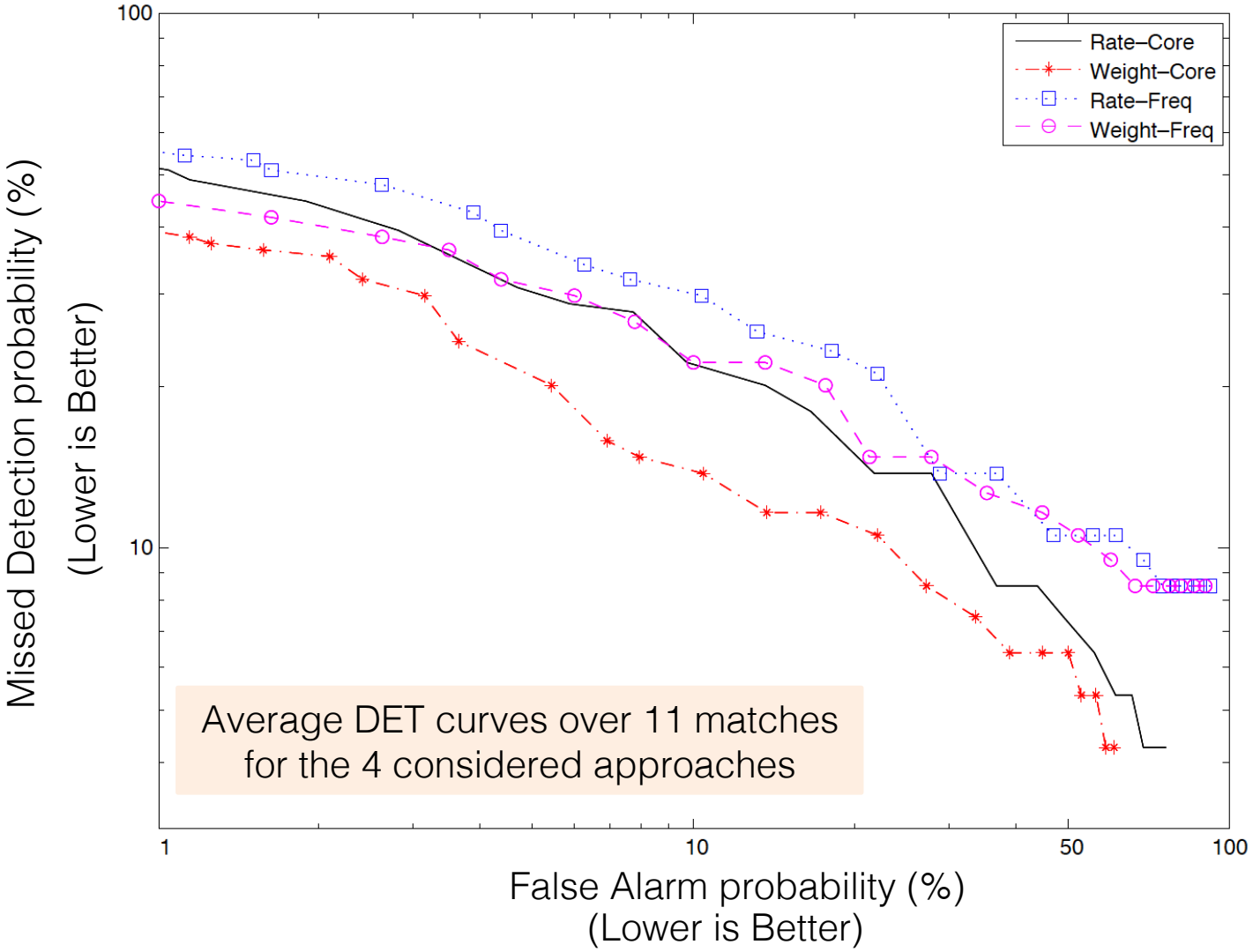
- ➢ Rate–Freq: the common baseline
- ➢ Rate–Core
- ➢ Weight–Core: Our approach
- ➢ Weight–Freq

Dataset

Match	#sub-events	#tweets
Germany - Argentina	8	1,907,999
Argentina - Belgium	7	1,355,472
France - Germany	6	1,321,781
Honduras-Switzerland	7	168,519
Greece - Ivory Coast	10	251,420
Croatia - Mexico	11	600,776
Cameroon - Brazil	11	532,756
Netherlands - Chile	7	301,067
Australia - Spain	9	252,086
Germany - Ghana	8	718,709
Australia - Netherlands	11	126,971
All Matches	95	7,537,556

FIFA 2014 World Cup Dataset

Evaluation (1/2)



Evaluation (2/2)

Method	Micro F1-score	Macro F1-score
Weight-Core	0.68	0.72
Rate-Core	0.61	0.63
Weight-Freq	0.61	0.64
Rate-Freq	0.54	0.60

Average micro and macro F1-score over 11 matches for the 4 considered approaches

Event type	#actual Events	#detected Events
Goal	32	30
Penalty	2	2
Red Card	1	0
Yellow Card	27	14
Match Start	11	8
Match End	11	11
Half Time	11	10

Number of sub-events detected

Tweet Summarization Performance

Time	Summary	ESPN FC
8’	Goal!!!!Argentina!! After eight minutes Argentina lead Belgium by 1-0 scored by Higuain	Goal! Argentina 1, Belgium 0. Gonzalo Higuain (Argentina) right footed shot from the centre of the box to the bottom left corner.
45’+2’	HT: Argentina 1-0 Belgium. Fantastic goal by Higuain gives Argentina the slight lead over the red devils.	First Half ends, Argentina 1, Belgium 0.
52’	52m - Belgium's Eden Hazard with the first yellow card of the game	Eden Hazard (Belgium) is shown the yellow card for a bad foul.
75’	Argentina 1 - 0 Belgium | Biglia booked a yellow card. Meanwhile, Chadli on for Eden Hazard.	Lucas Biglia (Argentina) is shown the yellow card for a bad foul.
90+5'	Well at least that goal makes them advance to the semi finals. Argentina gets the ticket to advance and Belgium goes home.	Match ends, Argentina 1, Belgium 0.

Summary of the Argentina vs. Belgium match generated automatically using Weight–Core and manually by ESPN

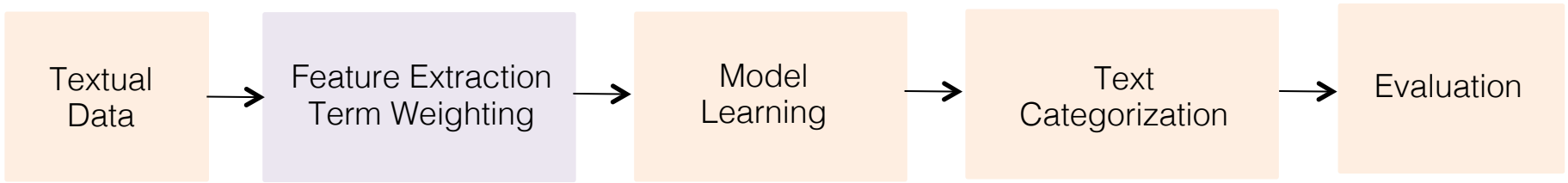
Summary

- Sub-event detection approach based on the k-core decomposition on graph-of-words
- The algorithm exploits the fact that the vocabulary of tweets gets more specific when a sub-event occurs
- The detection mechanism is able to accurately detect important moments as they occur
- The tweets selected by our system give an overview of the event

Tutorial Outline

- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

Text Categorization (TC) Pipeline



main focus

Applications of TC

- Applications of text classification are numerous:
  - News filtering
  - Document organization
  - Spam detection
  - Opinion mining

- Text documents classification compared to other domains:
  - High number of features
  - Sparse feature vectors
  - Multi-class scenario
  - Skewed class distribution

TC as a graph classification problem

TC as a Graph Classification Problem

- Single-label multi-class text categorization
- Graph-of-words representation of textual documents
- Mining of frequent subgraphs as features for classification
- Main core retention to reduce the graph's sizes
- Long-distance n-grams more discriminative than standard n-grams

[Rousseau et al., ACL '15]

Background (1/2)

- Text categorization
  - Standard baseline: unsupervised n-gram feature mining + supervised linear SVM learning
  - Common approach for spam detection: same with Naive Bayes

- n-grams to take into account some word order and some word dependence as opposed to unigrams

- Word inversion? Subset matching?

[Sebastiani, CSUR '02], [Aggarwal and Zhai, Mining Text Data '12]

Background (2/2)

- Graph classification
  - Subgraphs as features
  - Graph kernels [Vishwanathan et al., JMLR '10] [Covered next]
- Frequent subgraph feature mining
  - gSpan [Yan and Han, ICDM '02]
  - FFSM [Huan et al., ICDM '03]
  - Gaston [Nijssen and Kok, Elect. Notes TCS '04]
- Expensive to mine all subgraphs, especially for "large" collections of "large" graphs

- Unsupervised discriminative feature selection?

Subgraph-of-words

- A subgraph of size n corresponds to a long-distance n-gram
  - Takes into account word inversion and subset matching

- For instance, on the R8 dataset, {bank, base, rate} was a discriminative (top 5% SVM features) long-distance 3-gram for the category “interest”
  - “barclays **bank** cut its **base** lending **rate**”
  - “midland **bank** matches its **base rate**”
  - “**base rate** of natwest **bank** dropped”

Patterns hard to capture with traditional n-gram bag-of-words

Graph of Words Classification

Unsupervised feature mining and support selection

- gSpan mines the most frequent “subgraph-of-words” in the collection of graph-of-words
- Subgraph frequency == long-distance n-gram document frequency
- Minimum document frequency controlled via a support parameter
- The lower the support, the more features but the longer the mining, the feature vector generation and the learning
  - Unsupervised support selection using the elbow method (inspired from selecting the number of clusters in k-means)

[Rousseau et al., ACL ‘15]

Multiclass Scenario

- Text categorization ==

  multiple classes + skewed class distribution + single overall support value (local frequency)

- 100k features for majority classes vs. 100 features for minority ones

- Mining per class with same relative support value

Main Core Mining and n-gram Feature Selection

- Complexity to extract all features!
  - Reduce the size of the graphs
- Maintain word dependence and subset matching ⇒ keep the densest subgraphs

- Retain the main core of each graph-of-words use gSpan to mine frequent subgraphs in main cores
- Extract n-gram features on remaining text (terms in main cores)

Experimental Evaluation

- WebKB: 4 most frequent categories among labeled webpages from various CS departments – split into 2,803 for training and 1,396 for test [Cardoso-Cachopo, '07]
- R8: 8 most frequent categories of Reuters-21578, a set of labeled news articles from the 1987 Reuters newswire – split into 5,485 for training and 2,189 for test [Debole and Sebastiani, '05]
- LingSpam: 2,893 emails classified as spam or legitimate messages – split into 10 sets for 10-fold cross validation [Androutsopoulos et al., '00]
- Amazon: 8,000 product reviews over four different sub-collections (books, DVDs, electronics and kitchen appliances) classified as posi- tive or negative – split into 1,600 for training and 400 for test each [Blitzer et al., '07]

Models

- 3 baseline models (n-gram features)
  - kNN (k=5)
  - Multinomial Naive Bayes (similar results with Bernoulli)
  - Linear SVM

- 3 proposed approaches
  - gSpan + SVM (long-distance n-gram features)
  - MC + gSpan + SVM (long-distance n-gram features)
  - MC + SVM (n-gram features)

Evaluation Metrics

- Micro-averaged F1-score (accuracy, overall effectiveness)
- Macro-averaged F1-score (weight each class uniformly)
- Statistical significance of improvement in accuracy over the n-gram SVM baseline assessed using the micro sign test ($p < 0.05$)
- For the Amazon dataset, we report the average of each metric over the four sub-collections

Effectiveness Results (1/2)

Method \ Dataset	WebKB		R8	
	Accuracy	F1-score	Accuracy	F1-score
kNN (k=5)	0.679	0.617	0.894	0.705
NB (Multinomial)	0.866	0.861	0.934	0.839
linear SVM	0.889	0.871	0.947	0.858
gSpan + SVM	**0.912***	**0.882**	**0.955***	**0.864**
MC + gSpan + SVM	0.901*	0.871	0.949*	0.858
MC + SVM	0.872	0.863	0.937	0.849

Table: Test accuracy and macro-average F1-score. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using micro sign test with regards to the SVM baseline of the same column. gSpan mining support values are 1.6% (WebKB) and 7% (R8).

[Rousseau et al., ACL '15]

Effectiveness Results (2/2)

Method \ Dataset	LingSpam		Amazon	
	Accuracy	F1-score	Accuracy	F1-score
kNN (k=5)	0.910	0.774	0.512	0.644
NB (Multinomial)	0.990	0.971	0.768	0.767
linear SVM	**0.991**	**0.973**	0.792	0.790
gSpan + SVM	**0.991**	0.972	0.798*	0.795
MC + gSpan + SVM	0.990	**0.973**	**0.800***	**0.798**
MC + SVM	0.990	0.972	0.786	0.774

Table: Test accuracy and macro-average F1-score. Bold font marks the best performance in a column. * indicates statistical significance at $p < 0.05$ using micro sign test with regards to the SVM baseline of the same column. gSpan mining support values are 4% (LingSpam) and 0.5% (Amazon).

[Rousseau et al., ACL '15]

Dimension Reduction – Main Core

Dataset	# subgraphs before	# subgraphs after	reduction
WebKB	30,868	10,113	67 %
R8	39,428	11,373	71 %
LingSpam	54,779	15,514	72 %
Amazon	16,415	8,745	47 %

Table: Total number of subgraph features vs. number of subgraph features present only in main cores along with the reduction of the dimension of the feature space on all four datasets.

Unsupervised Support Selection

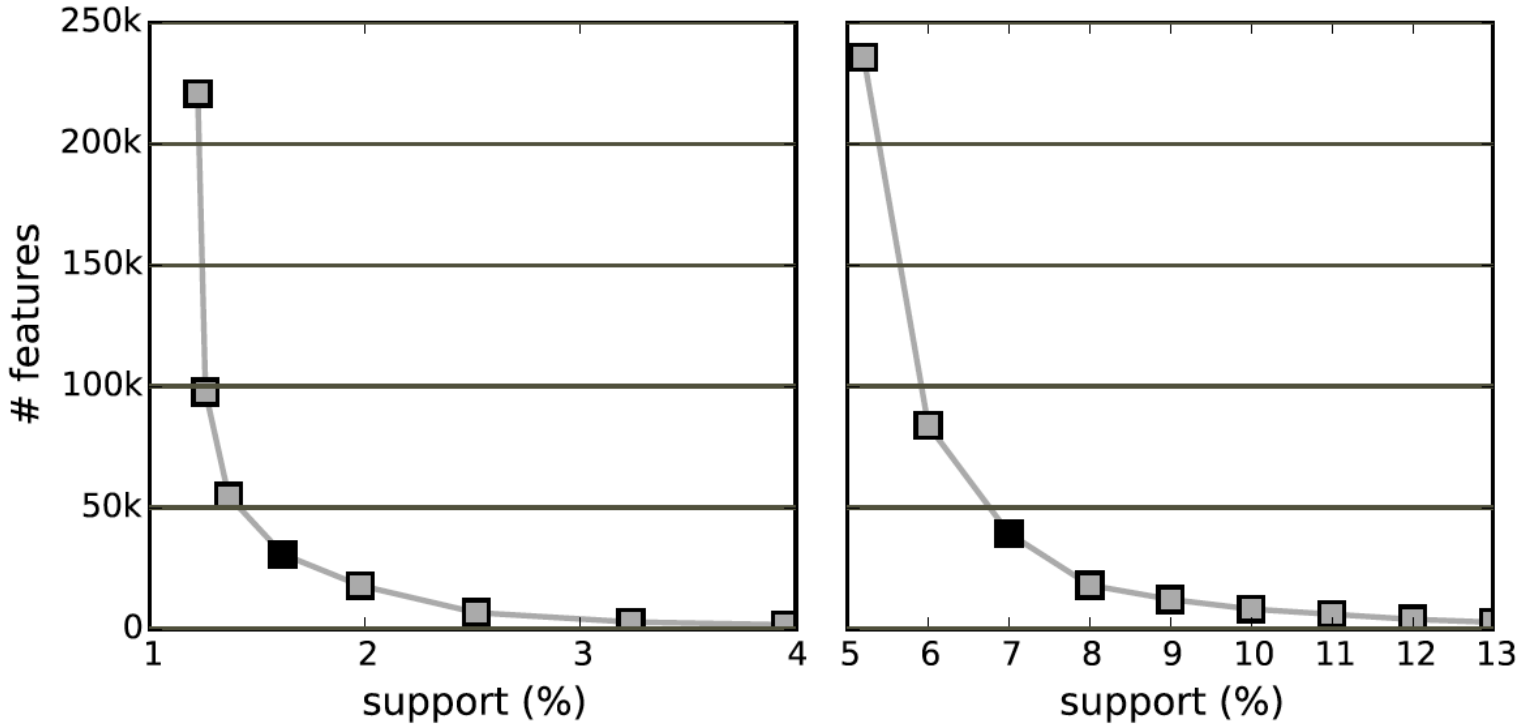


Figure: Number of subgraph features per support (%) on WebKB (left) and R8 (right) datasets. In black, the selected support chosen via the elbow method.

Distribution of Mined n-grams

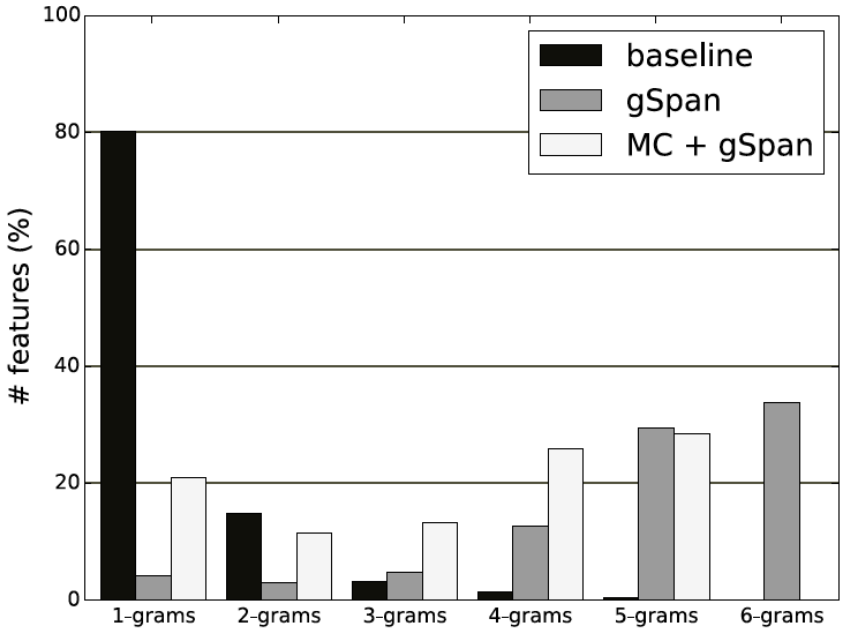


Figure: Distribution of n-grams (standard and long-distance ones) among all the features on WebKB dataset

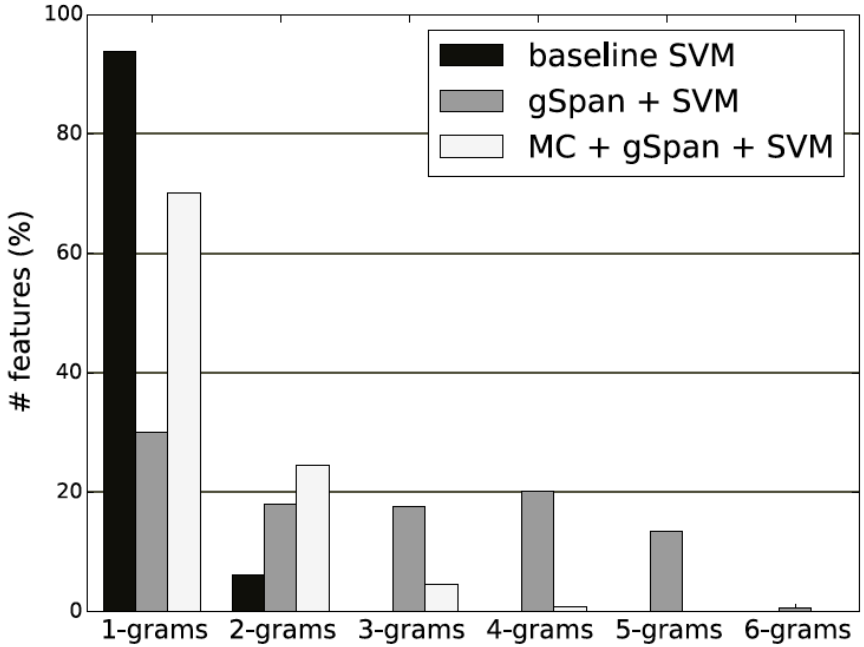


Figure: Distribution of n-grams (standard and long-distance ones) among the top 5% most discriminative features for SVM on WebKB dataset

Summary

- Explored a graph-based approach, to challenge the traditional bag-of-words for text classification
- First trained a classifier using frequent subgraphs as features for increased effectiveness
- Reduced each graph-of-words to its main core before mining the features for increased efficiency
- Reduced the total number of n-gram features considered in the baselines for little to no loss in prediction performances

Regularization for Text Categorization

Regularization for Text Categorization

- Why regularization?
  - Address overfitting: high training score, low test score
  - Better accuracy
- We want our model to generalize in new unseen test instances
- Harvest the full potential hidden in the rich textual data
- Feed meaningful group of words to group lasso for regularization

Objective Function + Loss

- Text categorization as a loss minimization problem:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underbrace{\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i)}_{\textbf{empirical risk}}$$

- Logistic regression with binary predictions ($y=\{-1,1\}$), $h_{\theta,b}(x) = \theta^T x+b$ and $L(x,\theta,y) = e^{-yh(x)}$ (log loss)
- Only minimizing the empirical risk can lead to overfitting.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \underbrace{\underbrace{\sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i)}_{\textbf{empirical risk}} + \underbrace{\lambda \Omega(\boldsymbol{\theta})}_{\textbf{penalty term}}}_{\textbf{expected risk}}$$

- L1, L2 regularization aka lasso [Tibshirani, '96] and ridge [Hoerl, Kennard, '70]

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i) + \lambda \sum_{j=1}^{p} |\boldsymbol{\theta}_j|$$

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}(\boldsymbol{x}^i, \boldsymbol{\theta}, y^i) + \lambda \sum_{j=1}^{p} \boldsymbol{\theta}_j{}^2$$

Learning

- A constrained optimization problem is formed that can be solved as an augmented Lagrangian problem:

$$\Omega_{las}(\boldsymbol{\theta}) + \Omega_{glas}(\mathbf{v}) + \mathcal{L}(\boldsymbol{\theta}) + \mathbf{u}^\top(\mathbf{v} - M\boldsymbol{\theta}) + \frac{\rho}{2}\|\mathbf{v} - M\boldsymbol{\theta}\|_2^2$$

- The problem becomes the iterative update of θ, v and u:

$$\min_{\boldsymbol{\theta}} \Omega_{las}(\boldsymbol{\theta}) + \mathcal{L}(\boldsymbol{\theta}) + \mathbf{u}^\top M\boldsymbol{\theta} + \frac{\rho}{2}\|\mathbf{v} - M\boldsymbol{\theta}\|_2^2$$

$$\min_{\mathbf{v}} \Omega_{glas}(\mathbf{v}) + \mathbf{u}^\top\mathbf{v} + \frac{\rho}{2}\|\mathbf{v} - M\boldsymbol{\theta}\|_2^2$$

$$\mathbf{u} = \mathbf{u} + \rho(\mathbf{v} - M\boldsymbol{\theta})$$

- [Yogatama and Smith '14] proved that ADMM for sparse overlapping group lasso converges. A good approximate solution is reached in a few tens of iterations

Structured Regularization

- In L1 and L2 regularization, features are considered as independent
- Group lasso: [Bakin, '99], [Yuan and Lin, '06] introduced group sparsity in the model:

$$\Omega(\boldsymbol{\theta}) = \lambda \sum_{g} \lambda_g \|\boldsymbol{\theta}_g\|_2$$

Algorithm 1 ADMM for overlapping group-lasso

Require: augmented Lagrangian variable ρ, regularization strengths λ_{glas} and λ_{las}

1: **while** update in weights not small **do**

2: $\quad \boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, \Omega_{las}(\boldsymbol{\theta}) + \mathcal{L}(\boldsymbol{\theta}) + \frac{\rho}{2} \sum_{i=1}^{V} N_i(\boldsymbol{\theta}_i - \mu_i)^2$

3: $\quad$ **for** $g = 1$ to G **do**

4: $\quad\quad \mathrm{v}_g = \text{prox}_{\Omega_{glas}, \frac{\lambda_g}{\rho}}(z_g)$

5: $\quad$ **end for**

6: $\quad u = u + \rho(\mathrm{v} - M\boldsymbol{\theta})$

7: **end while**

Structured Regularization in NLP

Statistical regularizers:

- Sentence regularizer (state-of-the-art) (overlapping)
  - Large numebr of groups but small group sizes.

Semantic regularizers:

- LDA and LSI regularizers
  - groups are considered the LDA/LSI results – keep top10 words in each group

Graph-of-words regularizers

- Graph-of-words regularizer: community detection on collection graph
- Word2vec regularizer: k-means clustering in word2vec space (overlapping)

Trying to extract groups of words that talk about similar topics.

[Skianis et al., EMNLP '16]

Results (1/2)

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	0.946	0.916	0.954	0.954	**0.968**	**0.968***	0.942	0.967*	**0.968***
	sports	0.908	0.907	0.925	0.920	0.959	0.964*	**0.966**	0.959*	0.946*
	religion	0.894	0.876	0.895	0.890	0.918	0.907*	**0.934**	0.911*	0.916*
	computer	0.846	0.843	0.869	0.856	0.891	0.885*	0.904	0.885*	**0.911***
Sentiment	vote	0.606	0.643	0.616	0.622	**0.658**	0.653	0.656	0.640	0.651
	movie	0.865	0.860	0.870	0.875	**0.900**	0.895	0.895	0.895	0.890
	books	0.750	0.770	0.760	0.780	0.790	0.795	0.785	0.790	**0.800**
	dvd	0.765	0.735	0.770	0.760	0.800	**0.805***	0.785	0.795*	0.795*
	electr.	0.790	0.800	0.800	**0.825**	0.800	0.815	0.805	0.820	0.815
	kitch.	0.760	0.800	0.775	0.800	0.845	**0.860***	0.855	0.840	0.855*

Table 2: Accuracy results on the test sets. Bold font marks the best performance for a dataset. * indicates statistical significance of improvement over lasso at $p < 0.05$ using micro sign test for one of our models LSI, GoW and word2vec (underlined).

	dataset	no reg.	lasso	ridge	elastic	group lasso				
						LDA	LSI	sentence	GoW	word2vec
20NG	science	100	1	100	63	19	20	86	19	21
	sports	100	1	100	5	60	11	6.4	55	44
	religion	100	1	100	3	94	31	99	10	85
	computer	100	2	100	7	40	35	77	38	18
Sentiment	vote	100	1	100	8	15	16	13	97	13
	movie	100	1	100	59	72	81	55	90	62
	books	100	3	100	14	41	74	72	90	99
	dvd	100	2	100	28	64	8	8	58	64
	electr.	100	4	100	6	10	8	43	8	9
	kitch.	100	5	100	79	73	44	27	75	46

Table 3: Fraction (in %) of non-zero feature weights in each model for each dataset: the smaller, the more compact the model.

Results (2/2)

	dataset	GoW	word2vec
20NG	science	79	691
	sports	137	630
	religion	35	639
	computer	95	594

Table 4: Number of groups.

	dataset	lasso	ridge	elastic	group lasso				
					LDA	LSI	sentence	GoW	word2vec
20NG	science	10	1.6	1.6	15	11	76	12	19
	sports	12	3	3	7	20	67	5	9
	religion	12	3	7	10	4	248	6	20
	computer	7	1.4	0.8	8	6	43	5	10

Table 5: Time (in seconds) for learning with best hyperparameters.

$= 0$	piscataway combination jil@donuts0.uucp jamie reading/seeing chambliss left-handedness abilities lubin acad sci obesity page erythromycin bottom
$\neq 0$	and space the launch health for use that medical you space cancer and nasa hiv health shuttle for tobacco that cancer that research center space hiv aids are use theory keyboard data telescope available are from system information space ftp

Table 6: Examples with LSI regularizer.

$= 0$	village town edc fashionable trendy trendy fashionable points guard guarding crown title champion champions
$\neq 0$	numbness tingling dizziness fevers laryngitis bronchitis undergo undergoing undergoes undergone healed mankind humanity civilization planet nasa kunin lang tao kay kong

Table 7: Examples with word2vec regularizer.

$= 0$	islands inta spain galapagos canary originated anodise advertises jewelry mercedes benzes diamond trendy octave chanute lillienthal
$\neq 0$	vibrational broiled relieving succumb spacewalks dna nf-psychiatry itself commented usenet golded insects alternate self-consistent retrospect

Table 8: Examples with graph-of-words regularizer.

Summary

- Find and extract semantic and syntactic structures that lead to sparser feature spaces → faster learning times
- Linguistic prior knowledge in the data can be used to improve categorization performance for baseline bag-of-words models, by mining inherent structures
- No significant change in results with different loss functions as the proposed regularizers are not log loss specific

Interesting questions

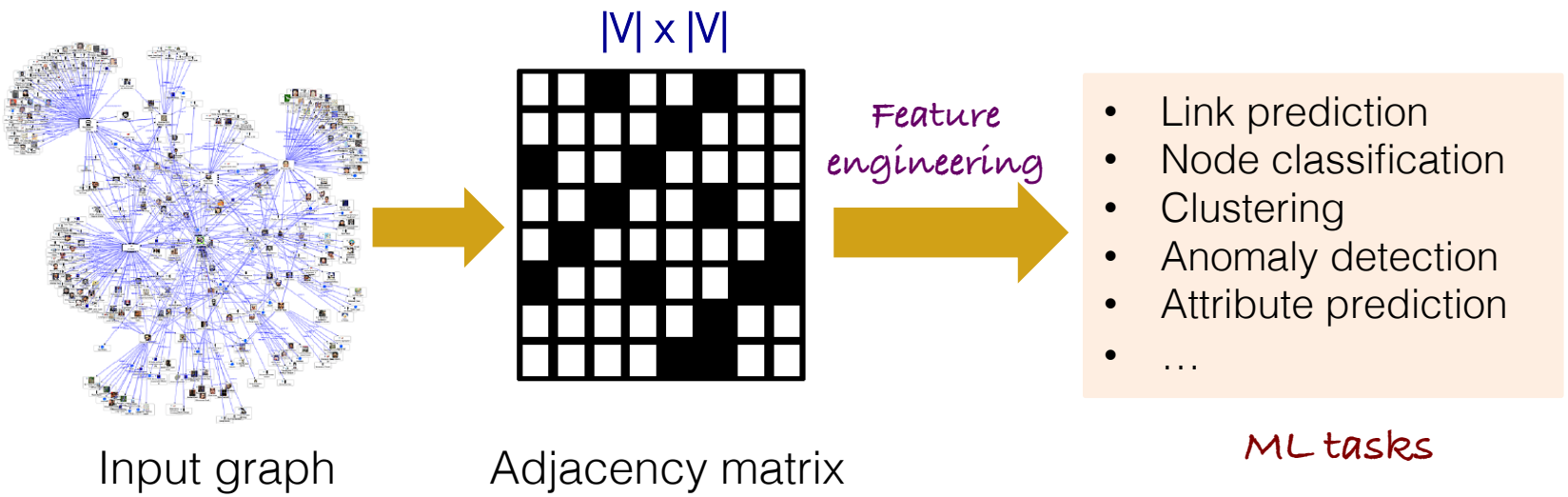
- How can we create and cluster graphs, i.e., covering weighted and/or signed cases?
- Find better clusters in word2vec? (+overlapping with GMM)
- Explore alternative regularization algorithms diverging from group-lasso?

Graph representation learning with applications in NLP

(text categorization and word analogy)

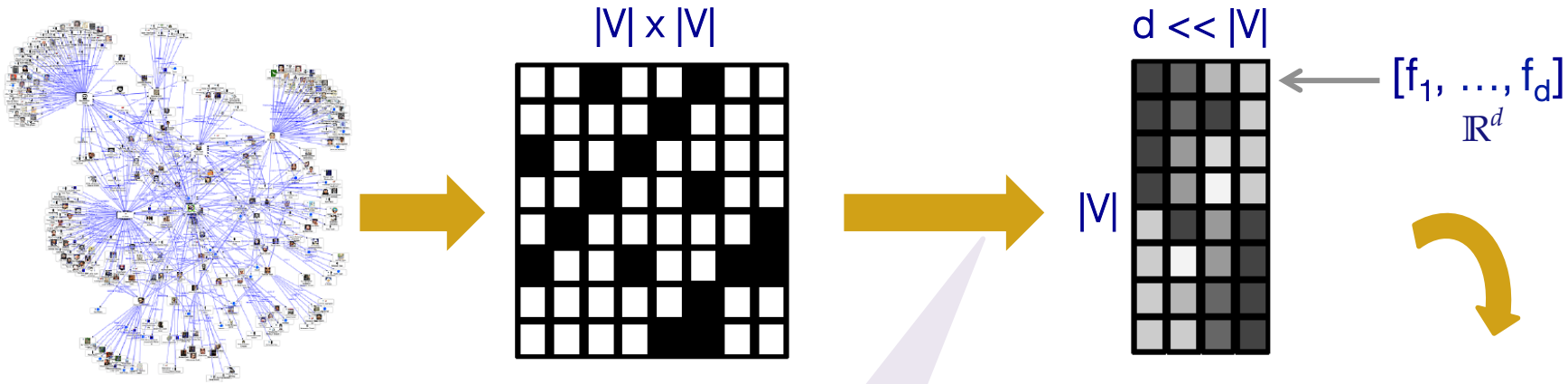
Feature Extraction From Graphs

- The first step of any ML algorithm for graphs is to extract graph features
  - Node features (e.g., degree)
  - Pairs of nodes (e.g., number of common neighbors)
  - Groups of nodes (e.g., community assignments)



Graph Representation

- Create features by transforming the graph into a lower dimensional latent representation

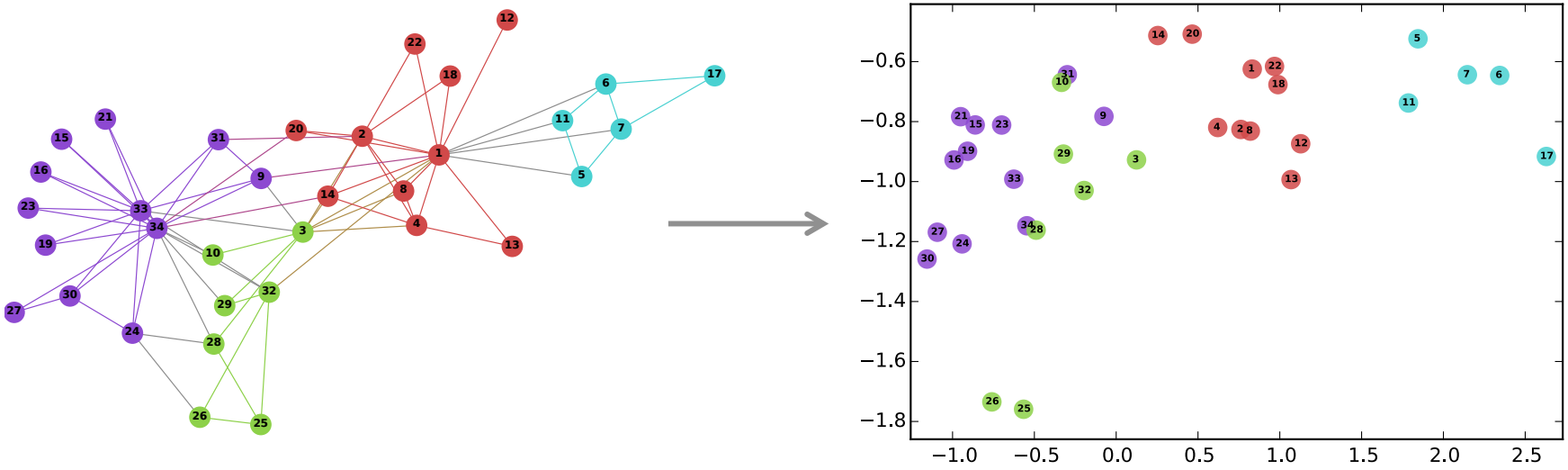


How to learn a latent representation of a graph?

- Link prediction
- Node classification
- Clustering
- Anomaly detection
- Attribute prediction
- …

ML tasks

Example: Community Detection



Input graph

Learn latent representation

Other applications: classification, link prediction, …

[Perozzi et al., KDD '14]

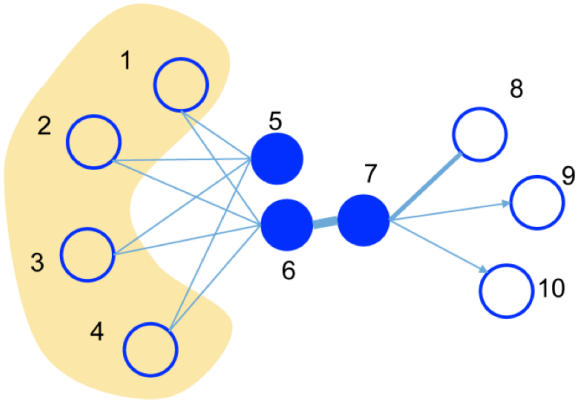
Problem Statement

- How to embed large networks into low-dimensional spaces?
- Requirements
  - Globality/Locality: It is desirable to preserve both local and global network structure when seeking for node representations
  - Scalability: When considering network with millions of nodes and billions of edges: traditional methods (nonlinear dimensionality reduction) suffer from lack of scalability

[Tang et al. '15]

Intuition – Local and Global Structures

- Local structure: observed edges in the network
  - First order proximity
  - Most traditional embedding methods (e.g., Isomap) capture first order proximity
- Global structure: nodes with shared neighbors are likely to be similar (homophily)



- Nodes 6 and 7: first-order proximity
  - Should be represented closely in the embedded space
- Nodes 5 and 6: second-order proximity
  - Same for those nodes

LINE algorithm: Form an objective function that optimizes both local and global network structure

[Tang et al., WWW '15]

LINE with First-order Proximity (1/2)

Model the probability of an edge (i, j) between v_i and v_j as

Embeddings space

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)}$$

$\vec{u}_i \in \mathbb{R}^d$ Low dimensional vector representation of node v_i

Joint probability between v_i and v_j

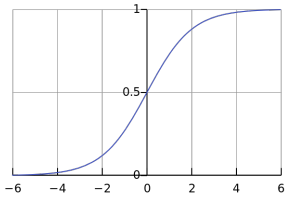
Original (graph) space

$$\hat{p}_1(i, j) = \frac{w_{ij}}{\sum_{(i,j) \in E} w_{ij}}$$

edge

Empirical distribution over the space V×V

Logistic function



Find vectors $\vec{u}_i \in \mathbb{R}^d$ to make those distributions to be as close as possible

LINE with First-order Proximity (2/2)

How to preserve first-order proximity?

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

Minimize the distance between two distributions

$$KL(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

KL-divergence

$$O_1 = KL(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

By finding those $\{\vec{u}_i\}_{i=1..|V|}$ that minimize O_1, we can represent every node in the d-dimensional space

LINE with Second-order Proximity (1/3)

- It assumes that nodes sharing many connections to other nodes are similar to each other
- Each node plays two roles:
  - The node itself
  - A specific "context" of other nodes

- For each node v_i, we model the conditional distribution $p_2(\bullet|v_i)$ over all the "contexts" (all the nodes in the network)
- Assumption of second-order proximity: Nodes with similar distributions $p_2(\bullet|v_i)$ over the "contexts" are similar

LINE with Second-order Proximity (2/3)

For directed edge (i, j), model the probability of context v_j generated by node v_i (i.e., probability of an edge from v_i to v_j)

Embeddings space

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}$$

$\vec{u}_i \in \mathbb{R}^d$ Low dimensional vector representation of node v_i

Conditional distribution $p_2(\bullet|v_i)$ over the contexts

Original (graph) space

$$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{d_i}$$

Out-degree of node i $d_i = \sum_{k \in N(i)} w_{ik}$

Empirical distribution over the space V×V

Make those distributions to be as close as possible

LINE with Second-order Proximity (3/3)

To preserve second-order proximity, minimize the distance between true and empirical distributions

$$O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i))$$

Minimize the distance between two distributions

- λ_i : represents the prestige of node i in the graph
- Set $\lambda_i = d_i$

KL-divergence

$$O_2 = -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i)$$

By finding those $\{\vec{u}_i\}_{i=1..|V|}$ and $\{\vec{u}'_i\}_{i=1..|V|}$ that minimize O_2, we can represent every node i with d-dimensional space $\vec{u}_i$

Combining Both Models

- Goal: Embed the networks by preserving both the first-order and second-order proximity

1. Train the LINE model for first-order proximity
2. Train the LINE model for second-order proximity

Train separately

- Then, concatenate the embeddings trained by the two methods for each node

Experiments - Datasets

Word co-occurrence network

	Language Network	Social Network	
Name	WIKIPEDIA	FLICKR	YOUTUBE
Type	undirected,weighted	undirected,binary	undirected,binary
|V|	1,985,098	1,715,256	1,138,499
|E|	1,000,924,086	22,613,981	2,990,443
Avg. degree	504.22	26.37	5.25
#Labels	7	5	47
#train	70,000	75,958	31,703

[Tang et al., WWW '15]

Experiments – Word Analogy

- Language network: word analogy
  - Find solution to ("China", "Beijing" → "France, "?")
  - Given word embeddings, find word d^* whose embedding u_d is closest to vector $u_{Beijing} - u_{China} + u_{France}$

"Paris"

Proximity in terms of cosine similarity

Algorithm	Semantic (%)	Syntactic (%)	Overall (%)	Running time
GF	61.38	44.08	51.93	2.96h
DeepWalk	50.79	37.70	43.65	16.64h
SkipGram	69.14	57.94	63.02	2.82h
LINE-SGD(1st)	9.72	7.48	8.50	3.83h
LINE-SGD(2nd)	20.42	9.56	14.49	3.94h
LINE(1st)	58.08	49.42	53.35	2.44h
LINE(2nd)	**73.79**	**59.72**	**66.10**	2.55h

Line (2nd) outperforms other embedding methods in the word analogy task

Experiments – Document Classification

Metric	Algorithm	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1	GF	79.63	80.51	80.94	81.18	81.38	81.54	81.63	81.71	81.78
	DeepWalk	78.89	79.92	80.41	80.69	80.92	81.08	81.21	81.35	81.42
	SkipGram	79.84	80.82	81.28	81.57	81.71	81.87	81.98	82.05	82.09
	LINE-SGD(1st)	76.03	77.05	77.57	77.85	78.08	78.25	78.39	78.44	78.49
	LINE-SGD(2nd)	74.68	76.53	77.54	78.18	78.63	78.96	79.19	79.40	79.57
	LINE(1st)	79.67	80.55	80.94	81.24	81.40	81.52	81.61	81.69	81.67
	LINE(2nd)	79.93	80.90	81.31	81.63	81.80	81.91	82.00	82.11	82.17
	LINE(1st+2nd)	**81.04****	**82.08****	**82.58****	**82.93****	**83.16****	**83.37****	**83.52****	**83.63****	**83.74****
Macro-F1	GF	79.49	80.39	80.82	81.08	81.26	81.40	81.52	81.61	81.68
	DeepWalk	78.78	79.78	80.30	80.56	80.82	80.97	81.11	81.24	81.32
	SkipGram	79.74	80.71	81.15	81.46	81.63	81.78	81.88	81.98	82.01
	LINE-SGD(1st)	75.85	76.90	77.40	77.71	77.94	78.12	78.24	78.29	78.36
	LINE-SGD(2nd)	74.70	76.45	77.43	78.09	78.53	78.83	79.08	79.29	79.46
	LINE(1st)	79.54	80.44	80.82	81.13	81.29	81.43	81.51	81.60	81.59
	LINE(2nd)	79.82	80.81	81.22	81.52	81.71	81.82	81.92	82.00	82.07
	LINE(1st+2nd)	**80.94****	**81.99****	**82.49****	**82.83****	**83.07****	**83.29****	**83.42****	**83.55****	**83.66****

- Classification of Wikipedia articles
  - Choose articles from 7 categories
- How to obtain the document vectors for classification?
  - Average of the corresponding word vector representations

Tutorial Outline

- Part I. Graph-theoretic concepts and graph-based text representation
- Part II. Information retrieval
- Part III. Keyword extraction and text summarization
- Part IV. Text categorization
- Part V. Final remarks and future research directions

Summary

- Graphs have been widely used as modeling tools in
  - NLP
  - Text Mining
  - Information Retrieval

- Goal of the tutorial
  - Presentation of recent methods that rely on graph-based text representations to deal with various tasks in NLP and IR
  - Focus on the graph-of-words model
  - Borrow ideas from the graph mining and network analysis field

Thank You! - Questions?

- Fragkiskos D. Malliaros

  University of California San Diego

  fmalliaros@ucsd.edu

  http://fragkiskos.me

- Michalis Vazirgiannis

  École Polytechnique, France

  mvazirg@lix.polytechnique.fr

  http://www.lix.polytechnique.fr/~mvazirg

Tutorial material: http://fragkiskosm.github.io/projects/graph_text_tutorial

References (1/4)

- Boudin, F., and Morin, E. 2013. Keyphrase Extraction for N-best reranking in multi-sentence compression. In North American Chapter of the Association for Computational Linguistics (NAACL).
- Mehdad, Y., Carenini, G., Tompa, F. W., and Ng, R. T. 2013. Abstractive meeting summarization with entailment and fusion. In Proc. of the 14th European Workshop on Natural Language Generation (pp. 136-146).
- Manu Aery and Sharma Chakravarthy. 2005. Infosift: Adapting graph mining techniques for text classification. In FLAIRS, pages 277–282.
- Ricardo A. Baeza-Yates and Berthier Ribeiro-Neto.1999. Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Roi Blanco and Christina Lioma. 2012. Graph-based term weighting for information retrieval. Inf. Retr., 15(1):54–92.
- Florian Boudin. 2013. A comparison of centrality measures for graph-based keyphrase extraction (ijcnlp ’13). In Sixth International Joint Conference on Natural Language Processing, pages 834–838.
- Adrien Bougouin, Florian Boudin, and B´eatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In Sixth International Joint Conference on Natural Language Processing (IJCNLP ’13), pages 543–551.
- Adrien Bougouin, Florian Boudin, and B´eatrice Daille. 2016. Keyphrase annotation with graph co-ranking. In 26th International Conference on Computational Linguistics (COLING ’16).
- Gunes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. Journal of Artificial Intelligence Research, pages 457–479.
- Filippova, K. 2010. Multi-sentence compression: Finding shortest paths in word graphs. In Proceedings of the 23rd International Conference on Computational Linguistics (pp. 322-330). Association for Computational Linguistics.

References (2/4)

- Maria Grineva, Maxim Grinev, and Dmitry Lizorkin. 2009. Extracting key terms from noisy and multitheme documents. In Proceedings of the 18th International Conference on World Wide Web (WWW '09) , pages 661–670. ACM.
- Samer Hassan, Rada Mihalcea, and Carmen Banea. 2007. Random-walk term weighting for improved text classification. In ICSC, pages 242–249.
- Chuntao Jiang, Frans Coenen, Robert Sanderson, and Michele Zito. 2010. Text classification using graph mining-based feature extraction. Knowl.- Based Syst., 23(4):302–308.
- Marina Litvak and Mark Last. 2008. Graphbased keyword extraction for single-document summarization. In Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization (MMIES '08), pages 17–24. Association for Computational Linguistics.
- Marina Litvak, Mark Last, Hen Aizenman, Inbal Gobits, and Abraham Kandel. 2011. Degext — a language-independent graph-based keyphrase extractor. In Elena Mugellini, Piotr S. Szczepaniak, Maria Chiara Pettenati, and Maria Sokhn, editors, Proceedings of the 7th Atlantic Web Intelligence Conference (AWIC '08), pages 121–130.
- Fragkiskos D. Malliaros and Konstantinos Skianis. 2015. Graph-based term weighting for text categorization. In ASONAM, pages 1473–1479. ACM.
- Alex Markov, Mark Last, and Abraham Kandel. 2007. Fast categorization of web documents represented by graphs. In Advances in Web Mining and Web Usage Analysis, volume 4811, pages 56–71.
- Polykarpos Meladianos, Giannis Nikolentzos, Franc¸ois Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2015. Degeneracy-based real-time subevent detection in twitter stream. In Ninth International AAAI Conference on Web and Social Media (ICWSM '15).

References (3/4)

- Rada Mihalcea and Paul Tarau. 2004. TextRank: bringing order into texts. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04). Association for Computational Linguistics.
- Francois Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: new approach to ad hoc ir. In Proceedings of the 22nd ACM international conference on Conference on Information & Knowledge Management (CIKM '13), pages 59–68. ACM.
- Francois Rousseau and Michalis Vazirgiannis. 2015. Main core retention on graph-of-words for singledocument keyword extraction. In European Conference on Information Retrieval (ECIR '15), pages 382–393. Springer.
- Francois Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In ACL (1), pages 1702– 1712. The Association for Computer Linguistics.
- Francois Rousseau. 2015. Graph-of-words: mining and retrieving text with networks of features. Ph.D. thesis, Ecole Polytechnique.
- Stephen B. Seidman. 1983. Network Structure and Minimum Degree. Social Networks, 5:269–287.
- Konstantinos Skianis, Francois Rousseau, and Michalis Vazirgiannis. 2016. Regularizing text categorization with clusters of words. In EMNLP, pages 1827– 1837. The Association for Computational Linguistics.
- Shashank Srivastava, Dirk Hovy, and Eduard H. Hovy. 2013. A walk-based semantically enriched tree kernel over distributed word representations. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13), pages 1411–1416.
- Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In Proceedings of the 24th International Conference on World Wide Web (WWW '15), pages 1067-1077

References (4/4)

- Antoine J.-P. Tixier, Fragkiskos D. Malliaros, and Michalis Vazirgiannis. 2016a. A graph degeneracybased approach to keyword extraction. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP '16), pages 1860–1870. The Association for Computational Linguistics.
- Antoine J.-P. Tixier, Konstantinos Skianis, and Michalis Vazirgiannis. 2016b. Gowvis: a web application for graph-of-words-based text visualization and summarization. In ACL. The Association for Computational Linguistics.
- S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph kernels. J. Mach. Learn. Res., 11:1201–1242.
- Jia Wang and James Cheng. 2012. Truss decomposition in massive networks. Proc. VLDB Endow., 5(9):812–823.
- Wei Wang, DiepBich Do, and Xuemin Lin. 2005. Term graph model for text classification. In Advanced Data Mining and Applications, volume 3584, pages 19–30.
- Rui Wang, Wei Liu, and Chris McDonald. 2015. Corpus-independent generic keyphrase extraction using word embedding vectors. In Workshop on Deep Learning for Web Search and Data Mining (DL-WSDM '15).
- Xifeng Yan and Jiawei Han. 2002. gSpan: Graphbased substructure pattern mining. In Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM '02).