

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

Fragkiskos D. Malliaros

CentraleSupélec and Inria

Polykarpos Meladianos, Michalis Vazirgiannis

École Polytechnique

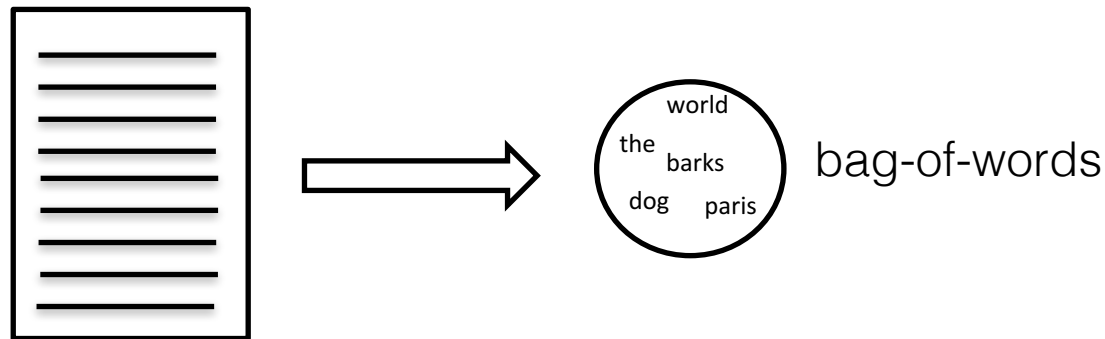
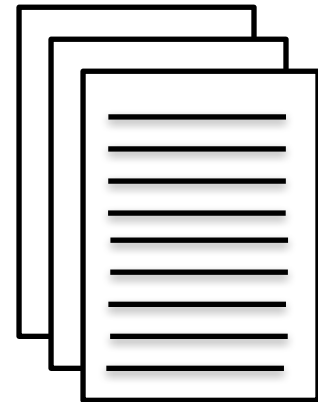
Updated slides at:

https://www.lix.polytechnique.fr/~mvazirg/gow_tutorial_webconf_2018.pdf

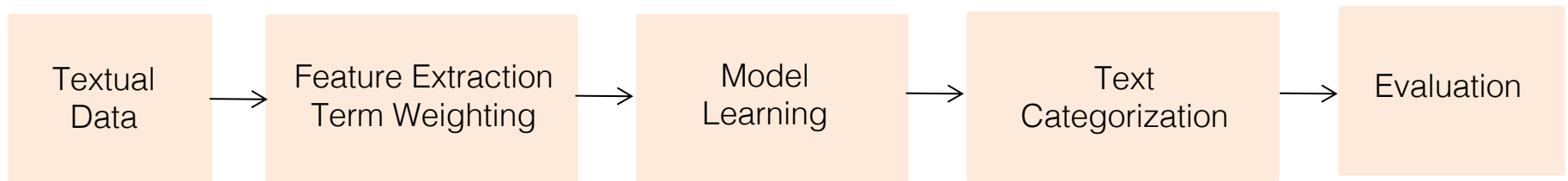
The WEB Conference
Lyon, France - April 23, 2018

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

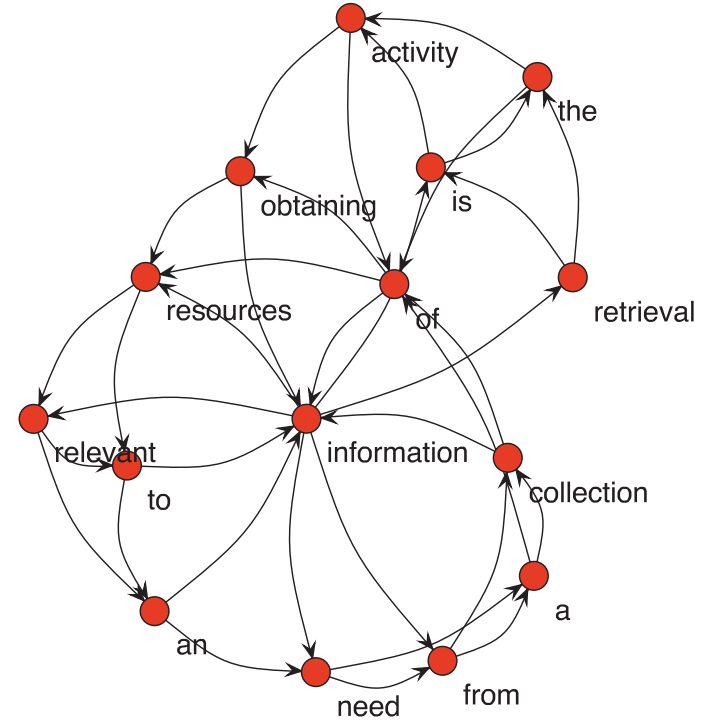
information retrieval is the activity of obtaining

information resources relevant to an information need

from a collection of information resources

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

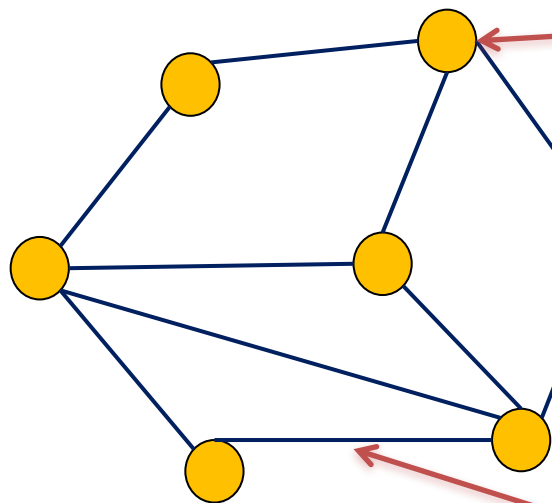
Tutorial Outline

- [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)

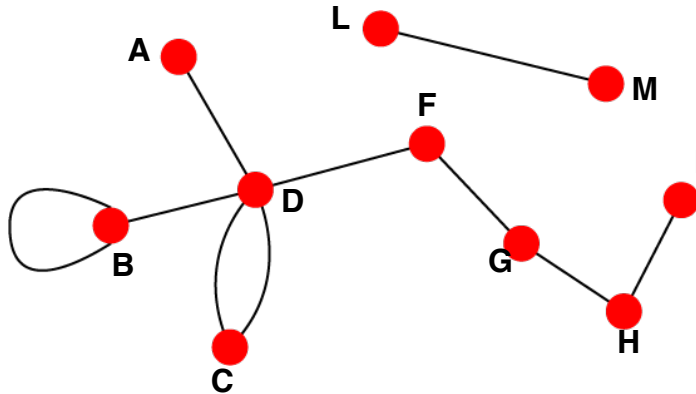
$G = (V, E)$
(network or graph)

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

Undirected vs. Directed Networks

Undirected

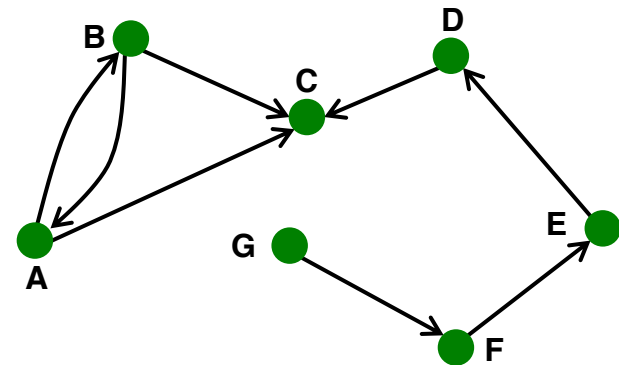
- **Links:** undirected (symmetrical, reciprocal)



- **Examples**
 - Collaborations
 - Friendship on Facebook

Directed

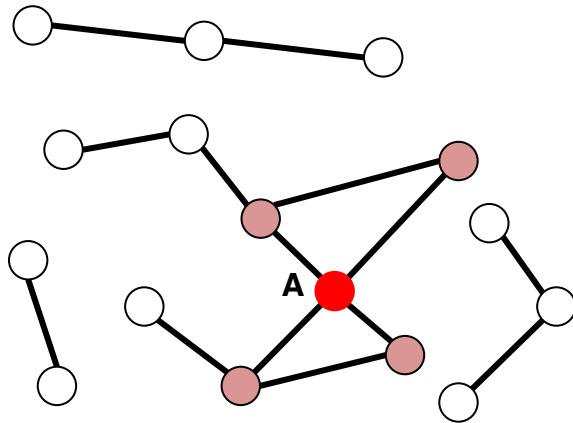
- **Links:** directed (arcs)



- **Examples**
 - Phone calls
 - Following on Twitter

Node Degree

Undirected

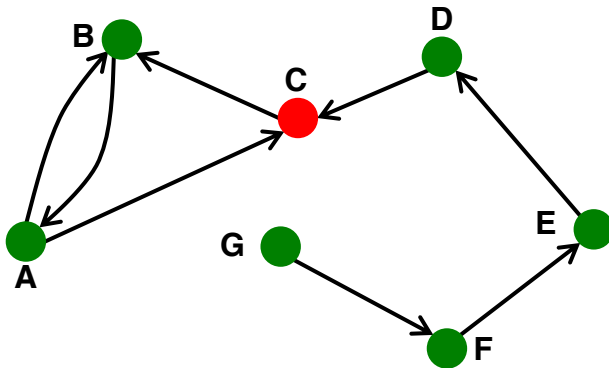


Node degree k_i : the number of edges adjacent to node i $k_A = 4$

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

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

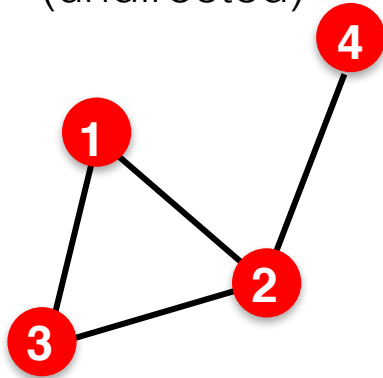
Source: Node with $k^{in} = 0$

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

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

More Types of Graphs

Unweighted
(undirected)



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

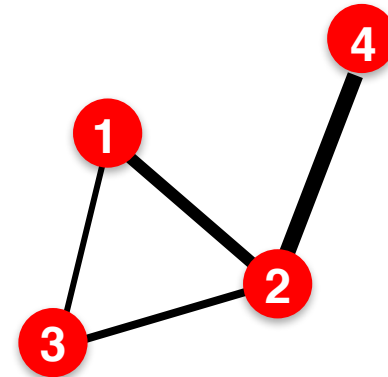
$$A_{ii} = 0$$

$$A_{ij} = A_{ji}$$

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

Examples: Friendship, Hyperlink

Weighted
(undirected)



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

$$A_{ii} = 0$$

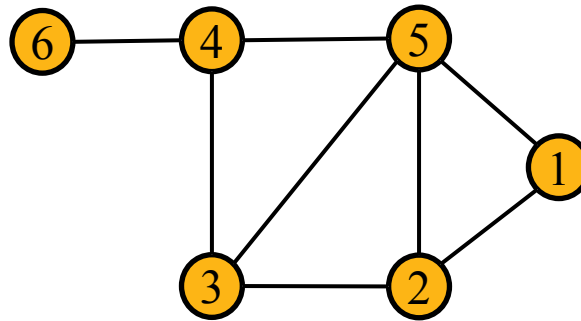
$$A_{ij} = A_{ji}$$

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

Examples: Collaboration, Internet, Roads

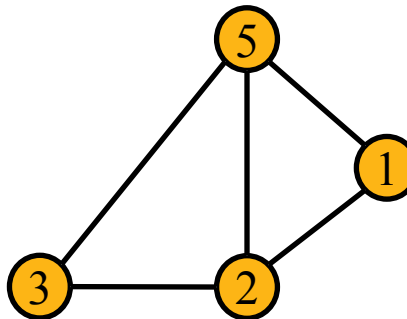
Subgraphs

- 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!

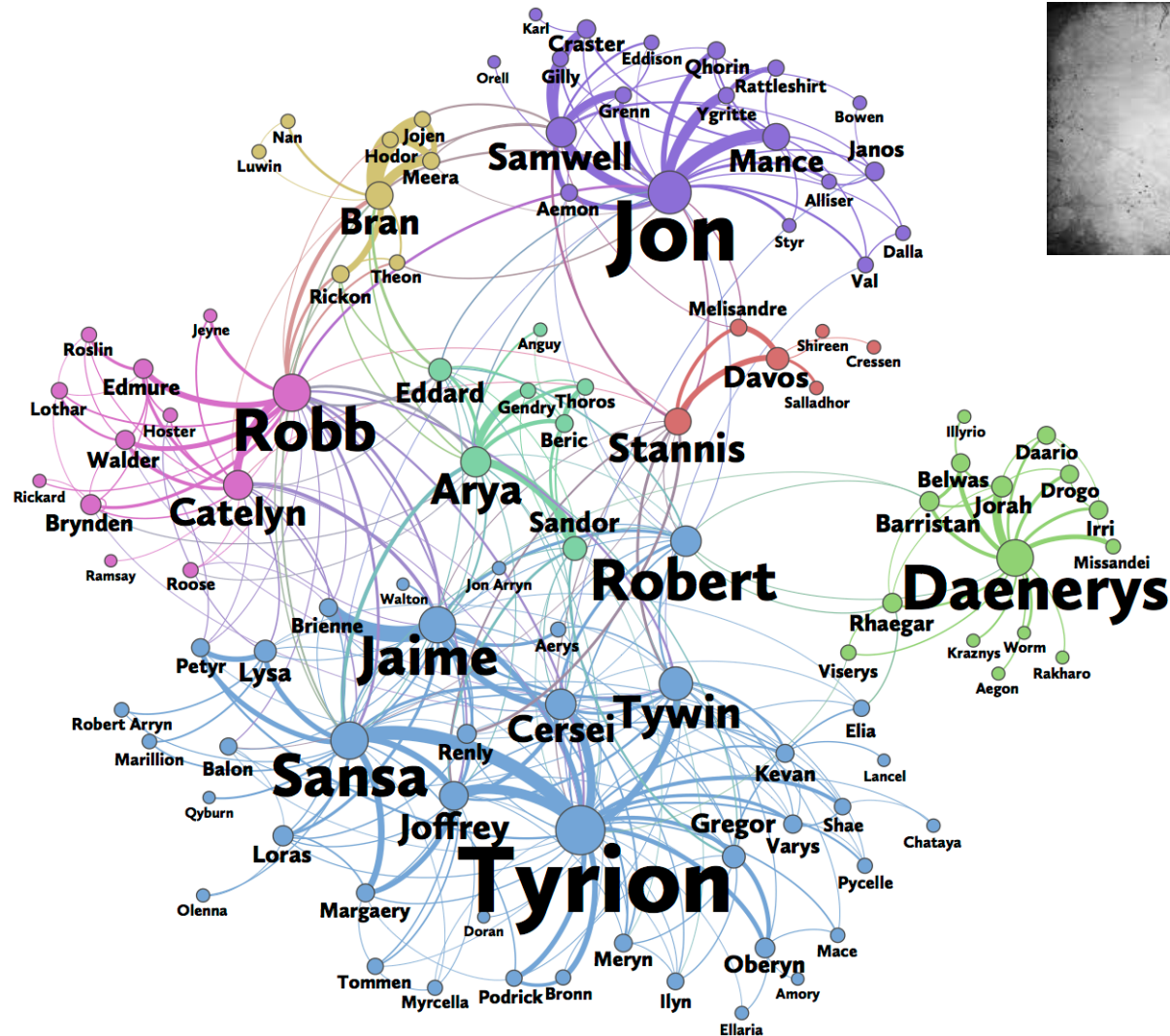
Choice of the proper network representation of a given system determines our ability to use networks meaningfully

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**

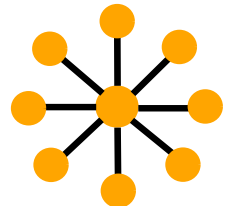
Centrality in Networks (2/2)



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

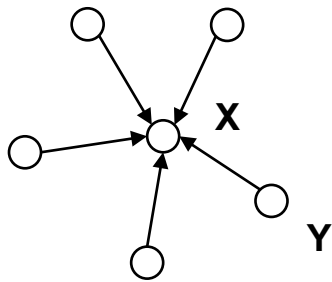


Measures of Centrality

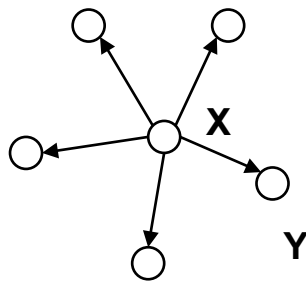
- This observation leads to the following classes of indices of centrality:
 - Measures based on **distances** (e.g., degree, closeness)
 - Measures based on **paths** (e.g., betweenness, Katz's index)
 - **Spectral** measures (eigenvector, PageRank, HITS, SALSA, random walks with restarts)
 - Measure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

A First Example

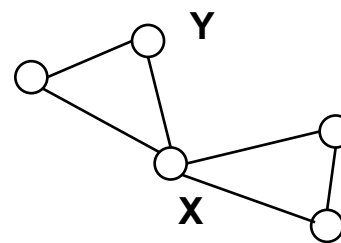
In each of the following networks, **X** has higher centrality than **Y** according to a particular measure



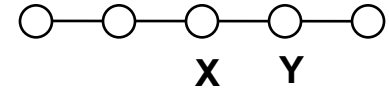
in-degree



out-degree



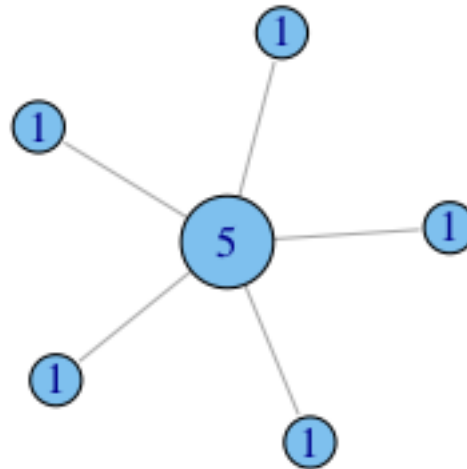
betweenness



closeness

Degree Centrality (1/2)

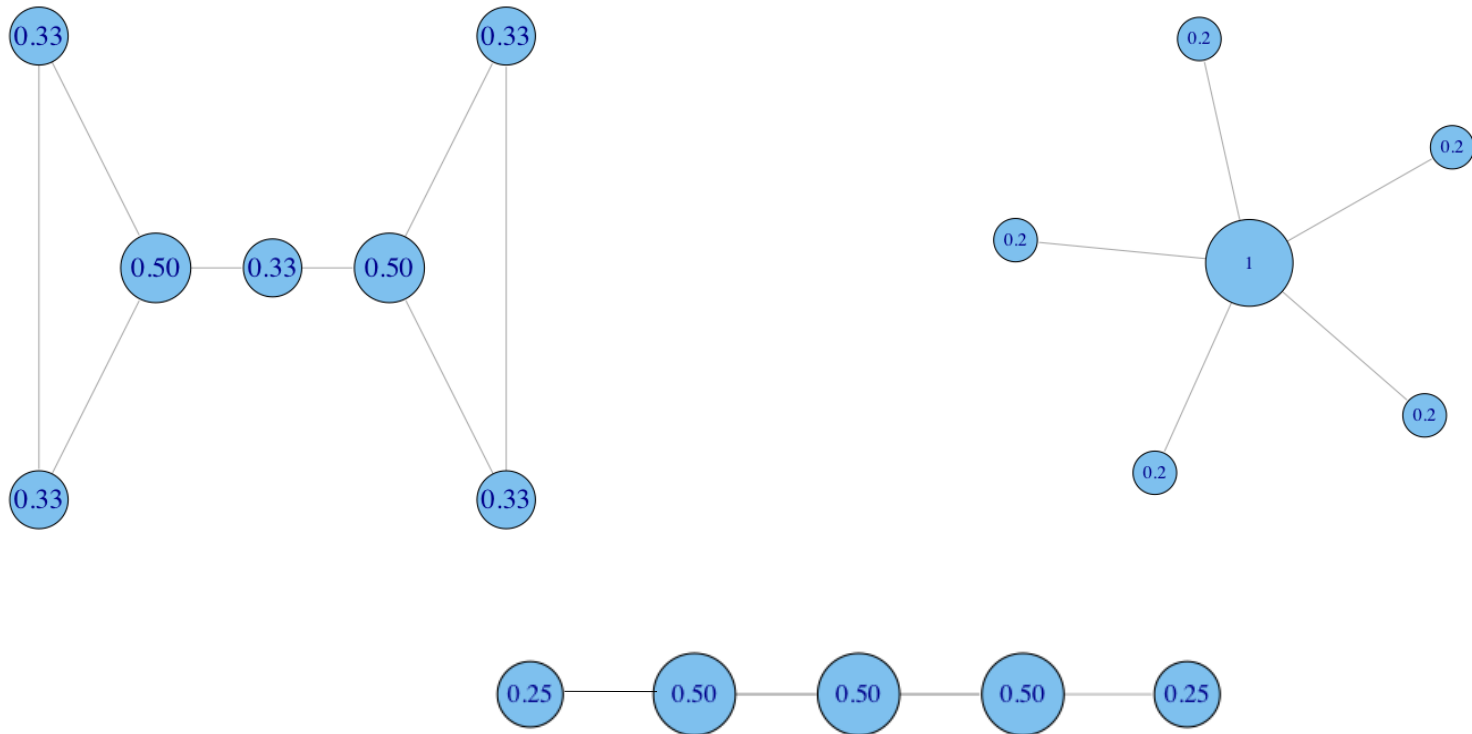
- **Idea:** A central node is one with many connections



- $C_d(i) = k(i)$, 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

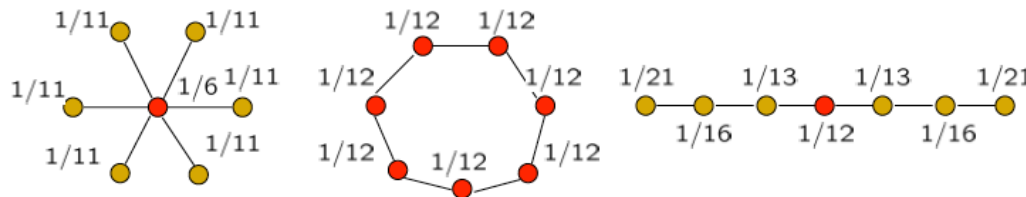
$$C_{cl}(i) = \frac{n-1}{\sum_{j \neq i} d(i, j)}$$

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 invert the distance?

- Nodes with **low mean distance** should get **high score**

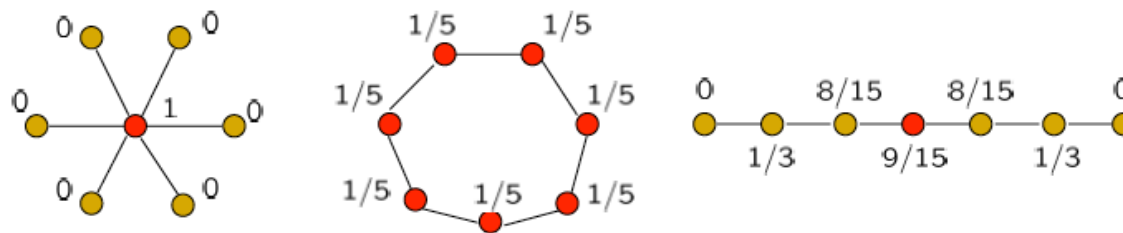
[Mateos, '17]

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
- $\sigma(s, t|i)$ 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: $\frac{C_{bt}(i)}{\binom{n-1}{2}}$

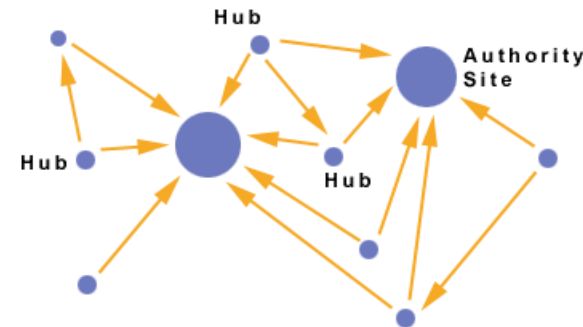
The HITS Algorithm

(Hubs and Authorities)

Hubs and Authorities (1/3)

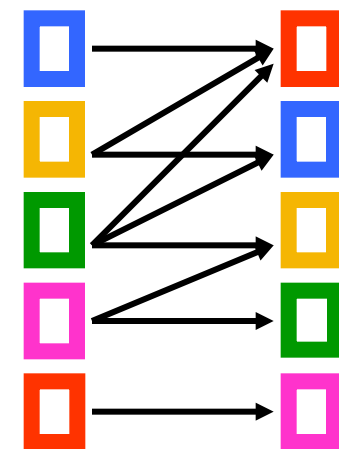
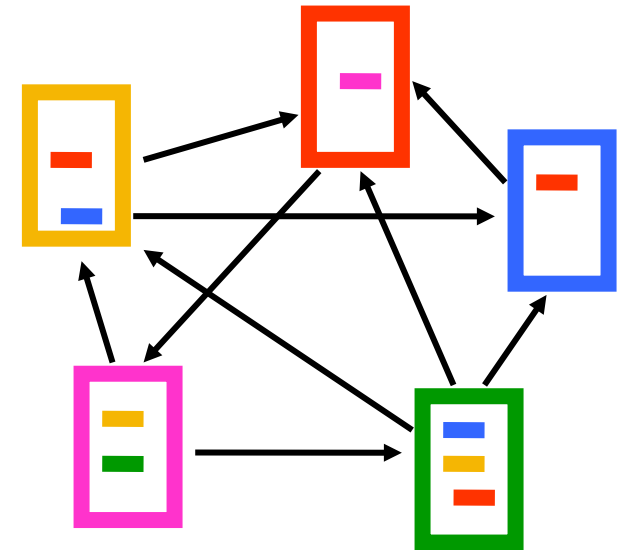
Interesting pages fall into two classes:

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

authorities

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

HITS and Eigenvectors

- HITS in vector notation
 - $\mathbf{a} = [\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n]^T$ and $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n]^T$
- We can rewrite \mathbf{a}_i and \mathbf{h}_i based on the adjacency matrix

$$h_i = \sum_{i \rightarrow j} \alpha_j \quad \text{as} \quad h_i = \sum_j A_{ij} \cdot \alpha_j$$

- Thus, $\mathbf{h} = \mathbf{A} \mathbf{a}$ and $\mathbf{a} = \mathbf{A}^T \mathbf{h}$

- $\mathbf{a}^{(t+1)} = \mathbf{A}^T \mathbf{h}^{(t)}$ and $\mathbf{h}^{(t+1)} = \mathbf{A} \mathbf{a}^{(t)}$
- $\mathbf{a}^{(t+1)} = \mathbf{A}^T \mathbf{A} \mathbf{a}^{(t)}$ and $\mathbf{h}^{(t+1)} = \mathbf{A} \mathbf{A}^T \mathbf{h}^{(t)}$

Repeated iterations
will converge to the
eigenvectors

SVD

- Authority weight vector \mathbf{a} : eigenvector of $\mathbf{A}^T \mathbf{A}$
- Hub weight vector \mathbf{h} : eigenvector of $\mathbf{A} \mathbf{A}^T$

The vectors \mathbf{a} and \mathbf{h} are
the singular vectors of

\mathbf{A}

PageRank

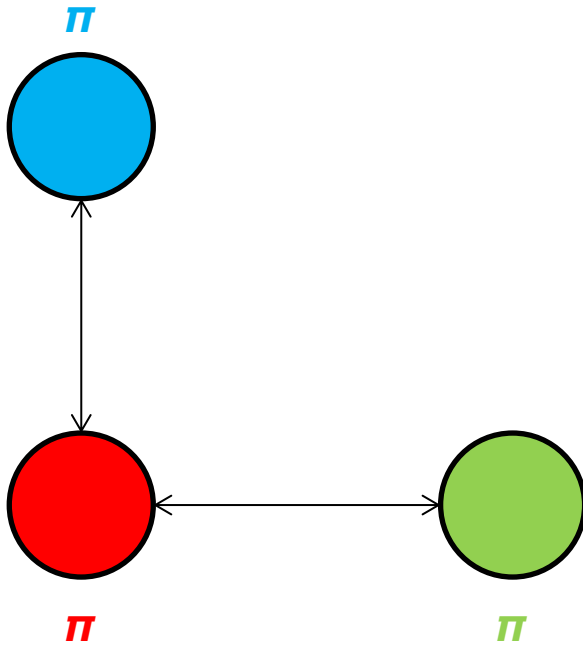
- **Good authorities** should be pointed by **good authorities**
 - The value of a node comes from the value of the nodes that point to it
- How do we implement that?
 - Assume that we have **a unit of authority** to distribute to all nodes
 - Initially, each node gets **$1/n$** amount of authority
 - 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

$$- \pi = \frac{1}{2} \quad \pi = \frac{1}{4} \quad \pi = \frac{1}{4}$$

A More Complex Example

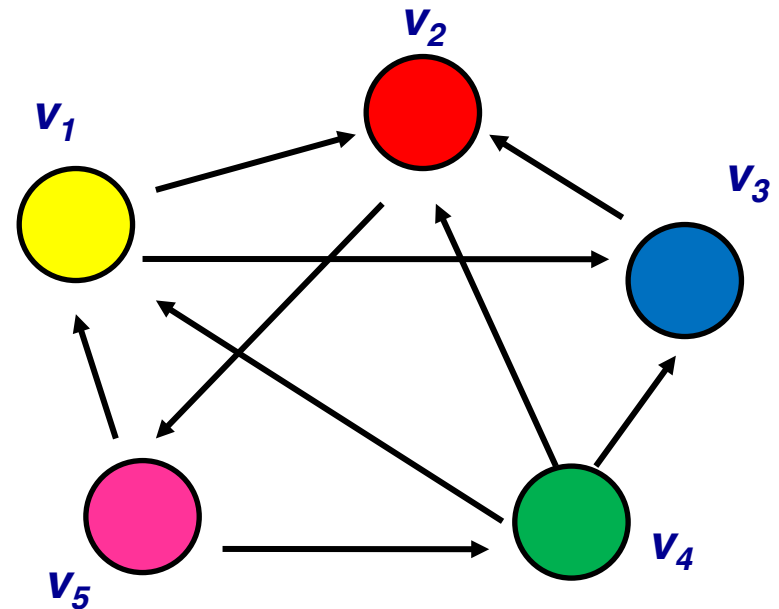
$$\pi_1 = 1/3 \pi_4 + 1/2 \pi_5$$

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

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

$$\pi_4 = 1/2 \pi_5$$

$$\pi_5 = \pi_2$$



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

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

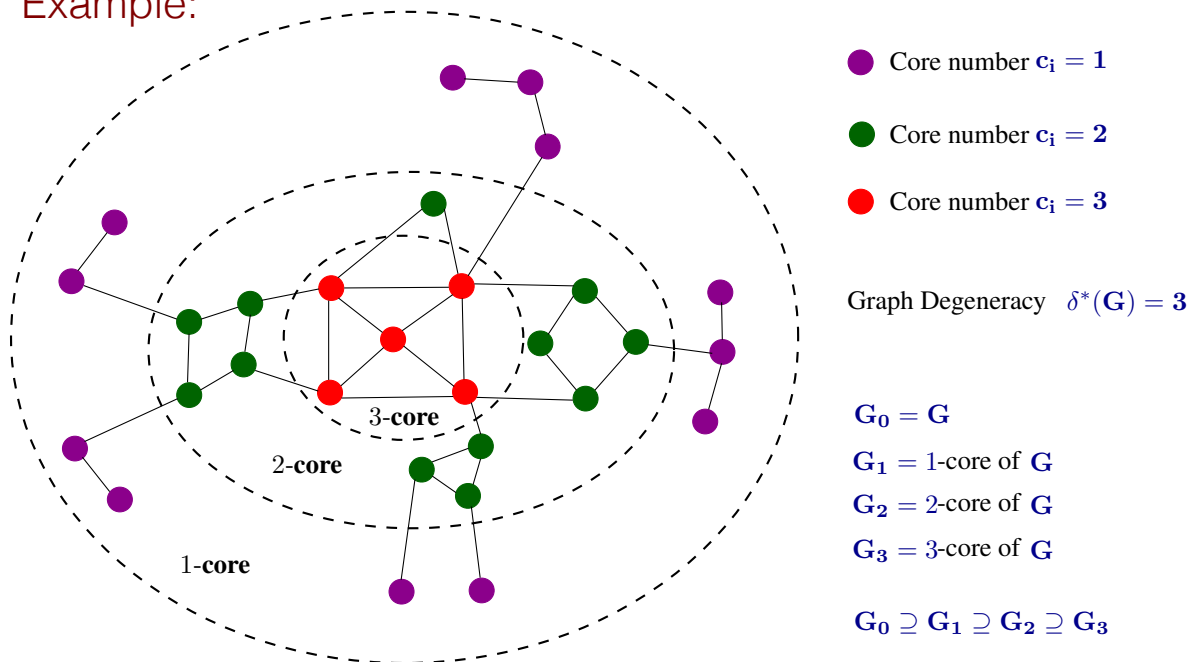
Core Decomposition

- 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:



Important property:

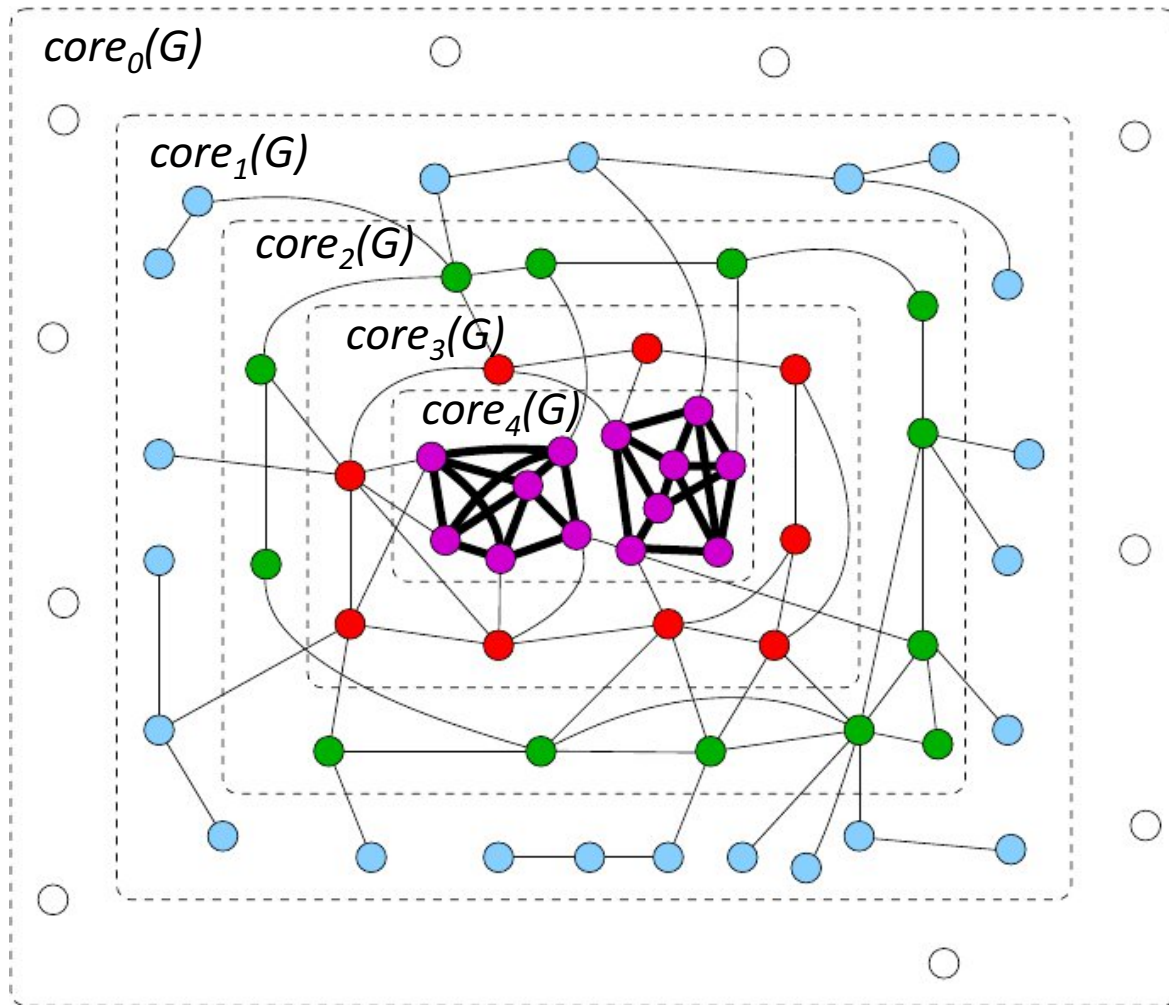
- Fast and easy to compute
- Linear to the size of the graph
- Scalable to large scale graphs

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 $k\text{-core}(G, k)$

Input: An undirected graph G and positive integer k

Output: $k\text{-core}(G)$

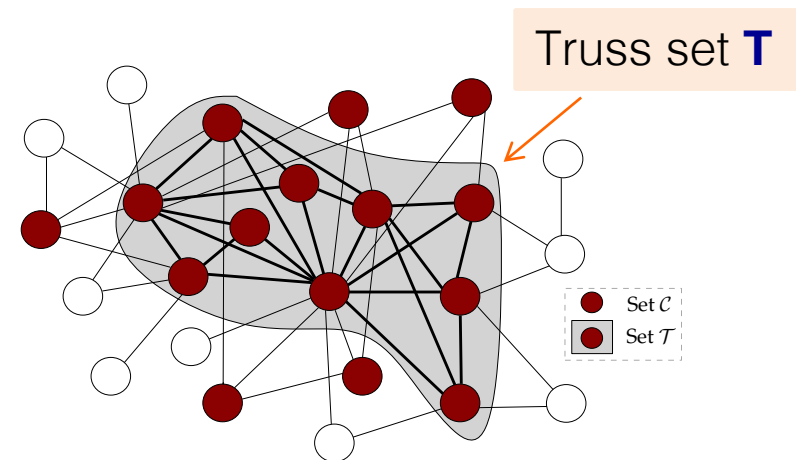
1. let $F := G$
2. while there is a node x in F such that $\deg_F(x) < k$
 delete node x from F
3. return F

- 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

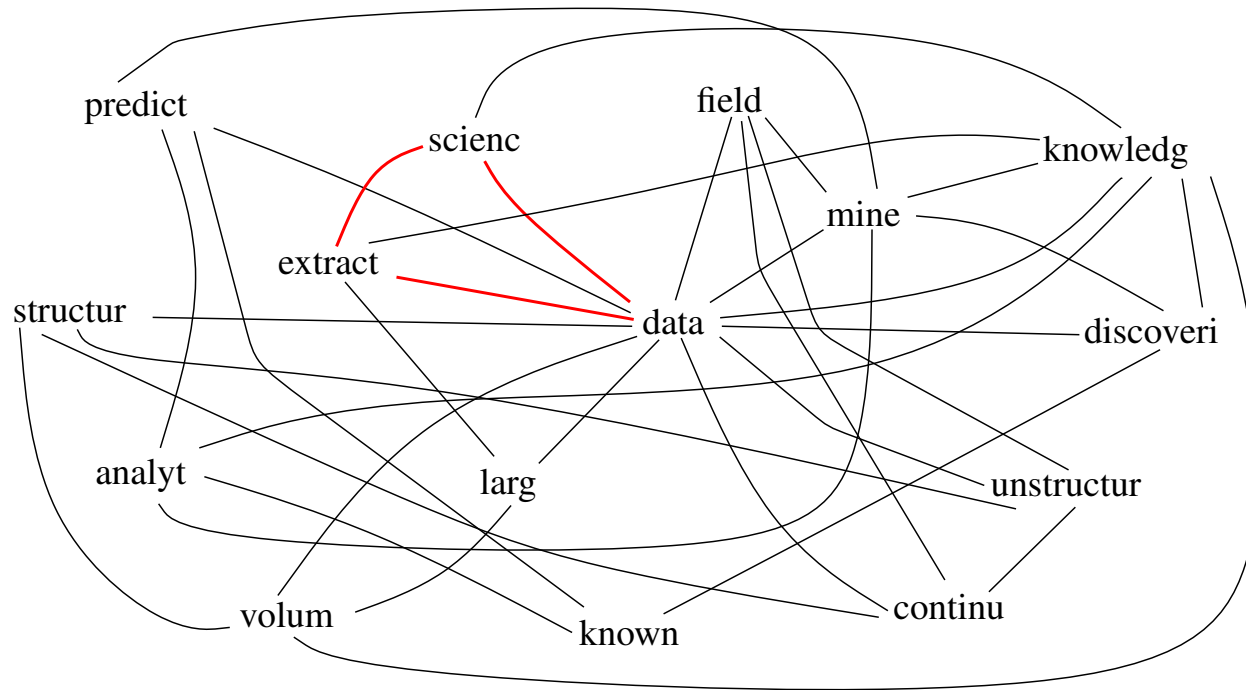
- Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be the graph that corresponds to a document \mathbf{d}
- The **nodes** can correspond to:
 - Paragraphs
 - Sentences
 - Phrases
 - Words [Main focus of the tutorial]
 - Syllables
- The **edges** of the graph can capture various types of relationships between 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 $\mathbf{d} \in \mathbf{D}$ is represented by a graph $\mathbf{G}_d = (\mathbf{V}_d, \mathbf{E}_d)$, where the nodes correspond to the terms \mathbf{t} of the document and the edges capture co-occurrence relationships between terms within a fixed-size sliding window of size \mathbf{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 \mathbf{w} of the sliding window**
 - Add edges between the terms of the document that co-occur within a sliding window of size \mathbf{w}
 - Larger window sizes produce graphs that are relatively dense

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.



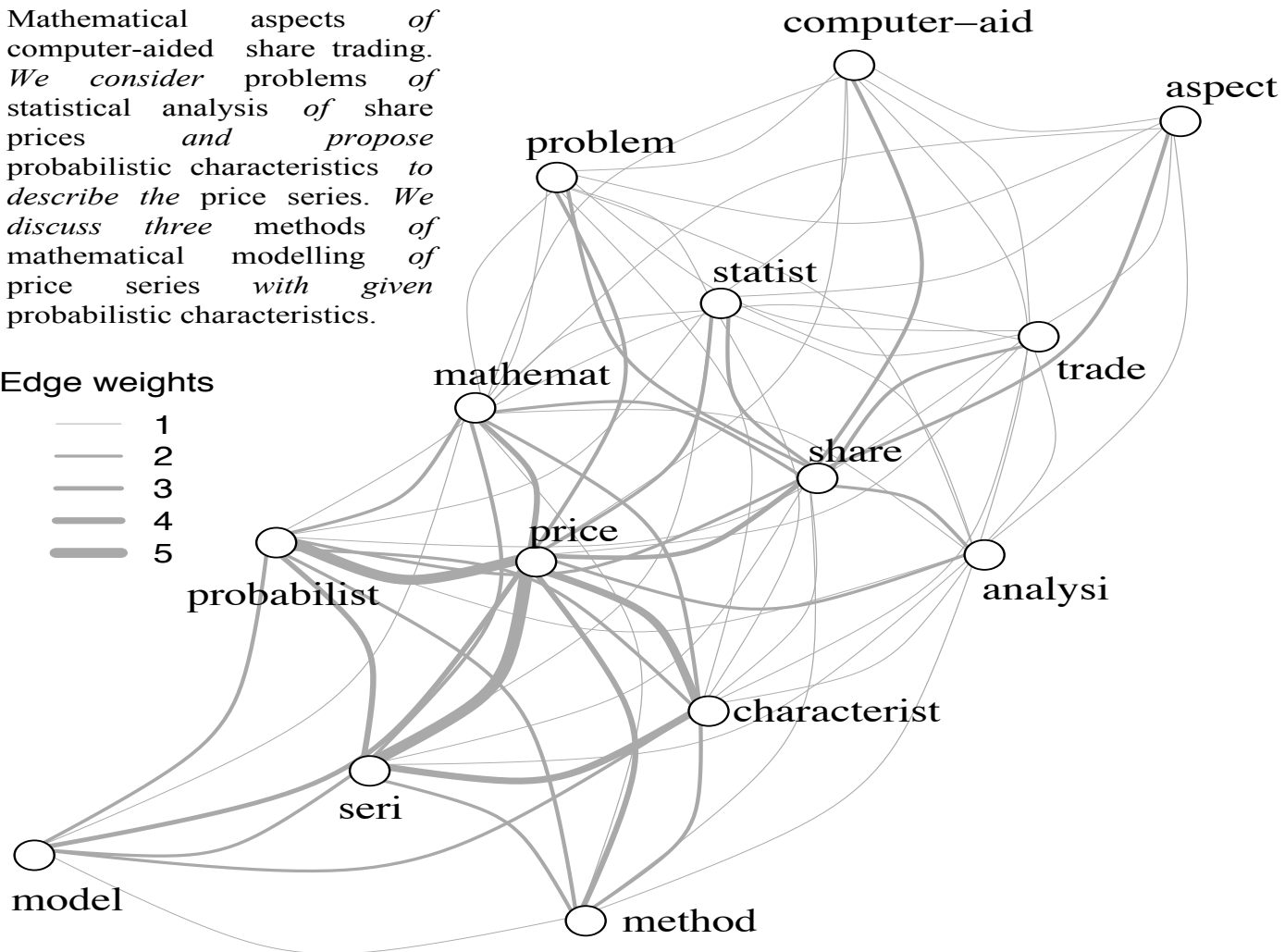
w = 3

unweighted, undirected graph

Example of Weighed Undirected GoW

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.

Edge weights



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

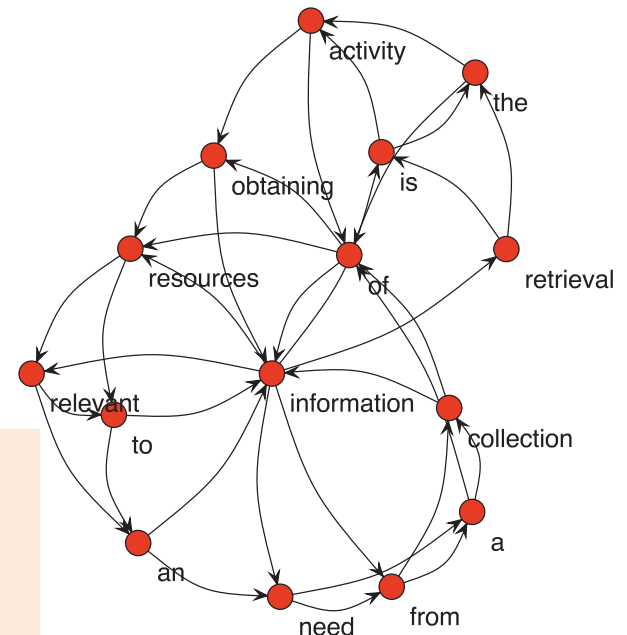
- The weight of a term in a document is its **in-degree in the graph-of-words**
- It represents the number of **distinct contexts** of occurrence
- We store the document as a vector of weights in the direct index and similarly in the inverted index

- 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 **ldl**, average document length **avdl**, asymptotical marginal gain k_1 (1.2), slope parameter **b**

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

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

- 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 **ldl**, average document length **avdl**, asymptotical marginal gain **k₁** (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)

Statistic	Dataset	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

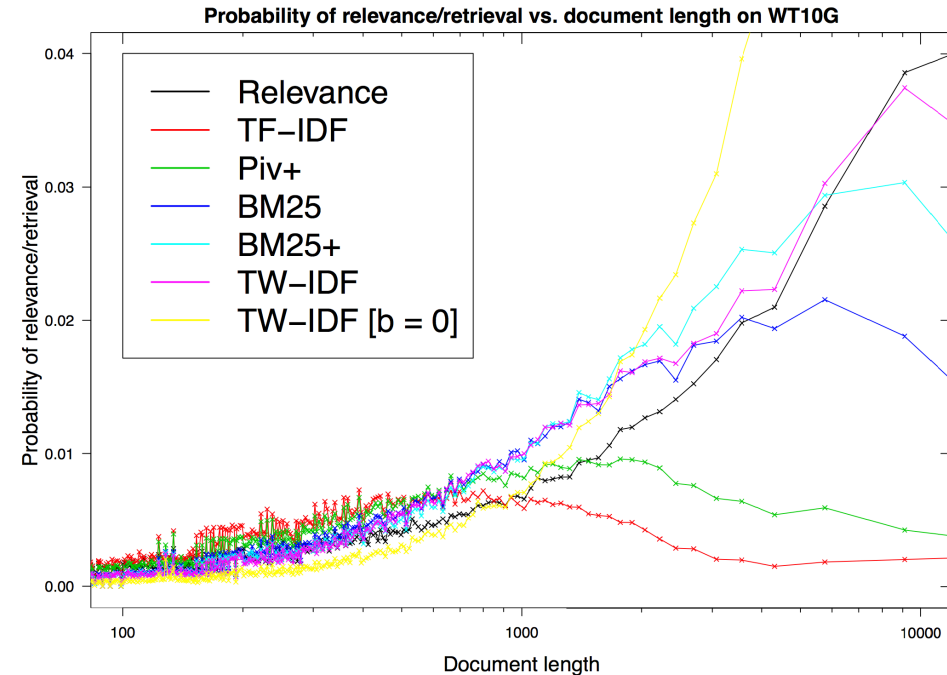
Table: Statistics on the four TREC datasets used; Disks 4&5 excludes the Congressional Record. The average values are computed per document.

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]

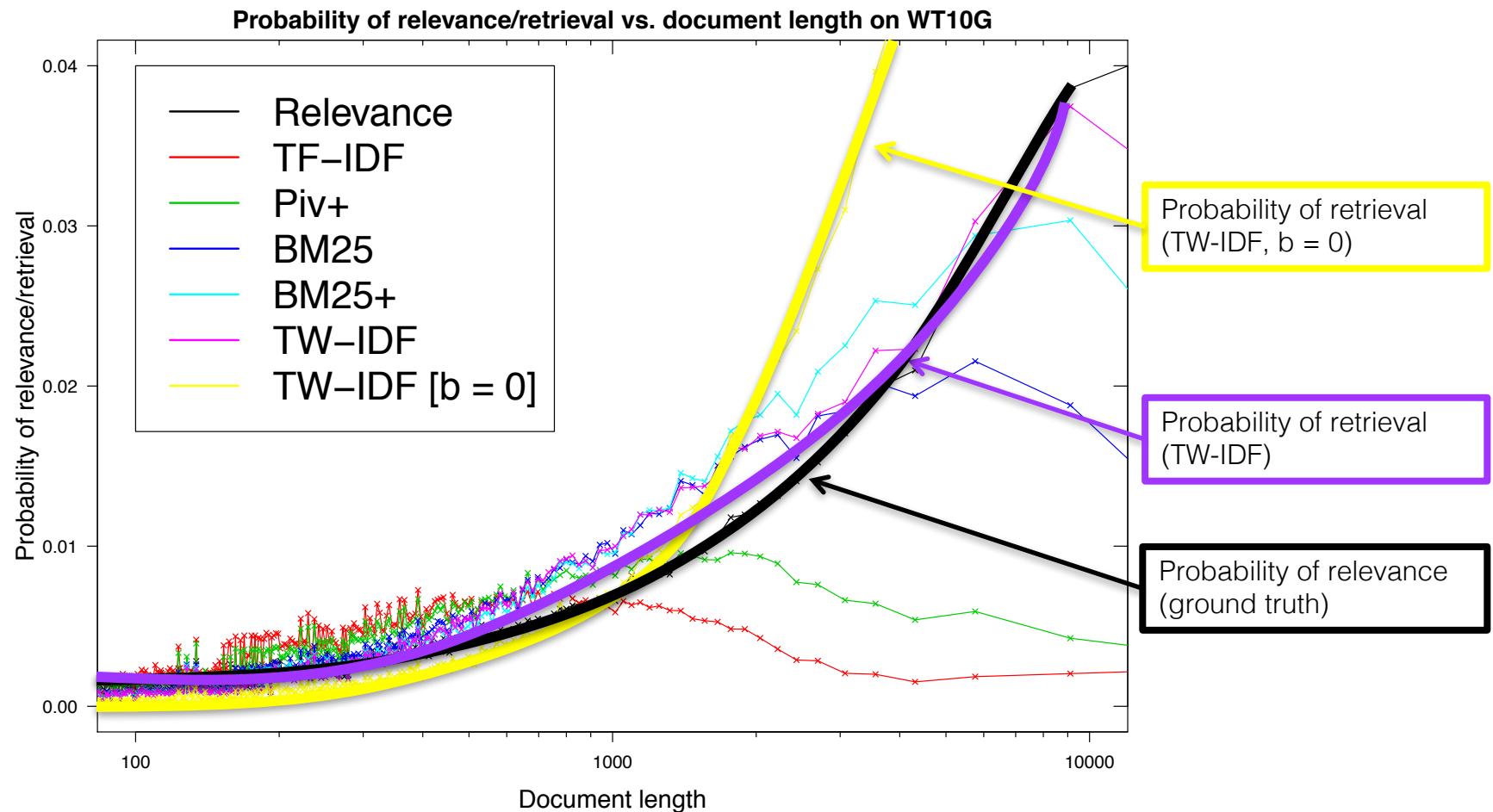
Graph-based Ad Hoc IR

- 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

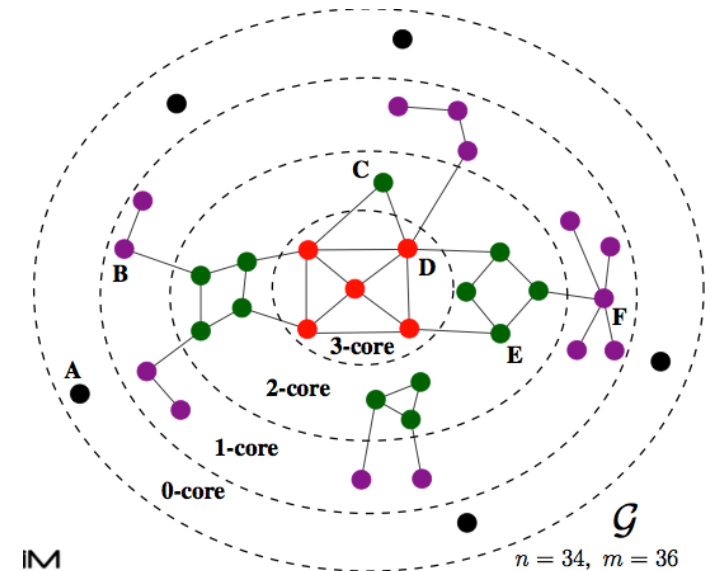
Applications are numerous

- **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]



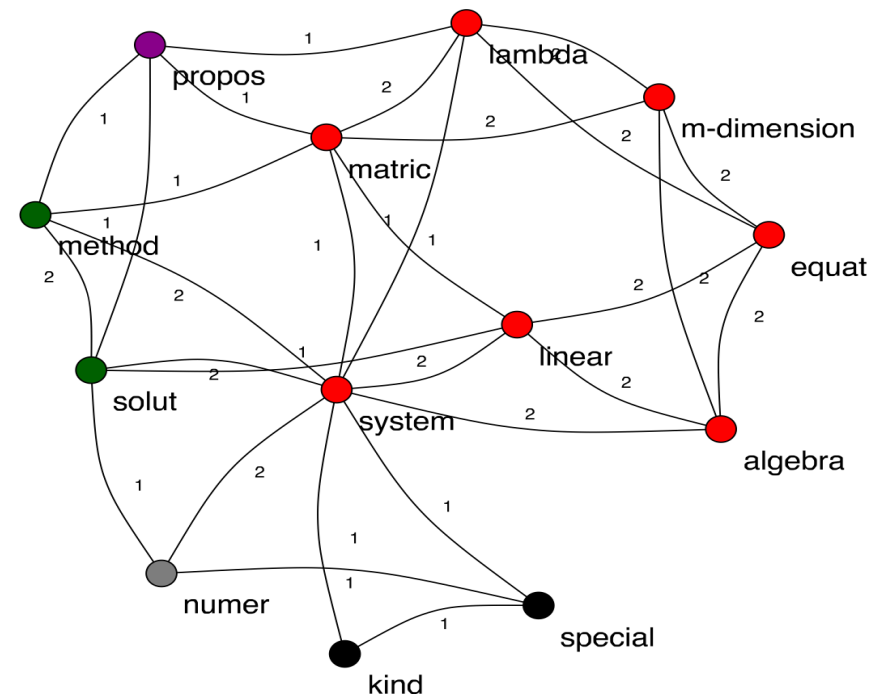
k-core decomposition of the graph

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)

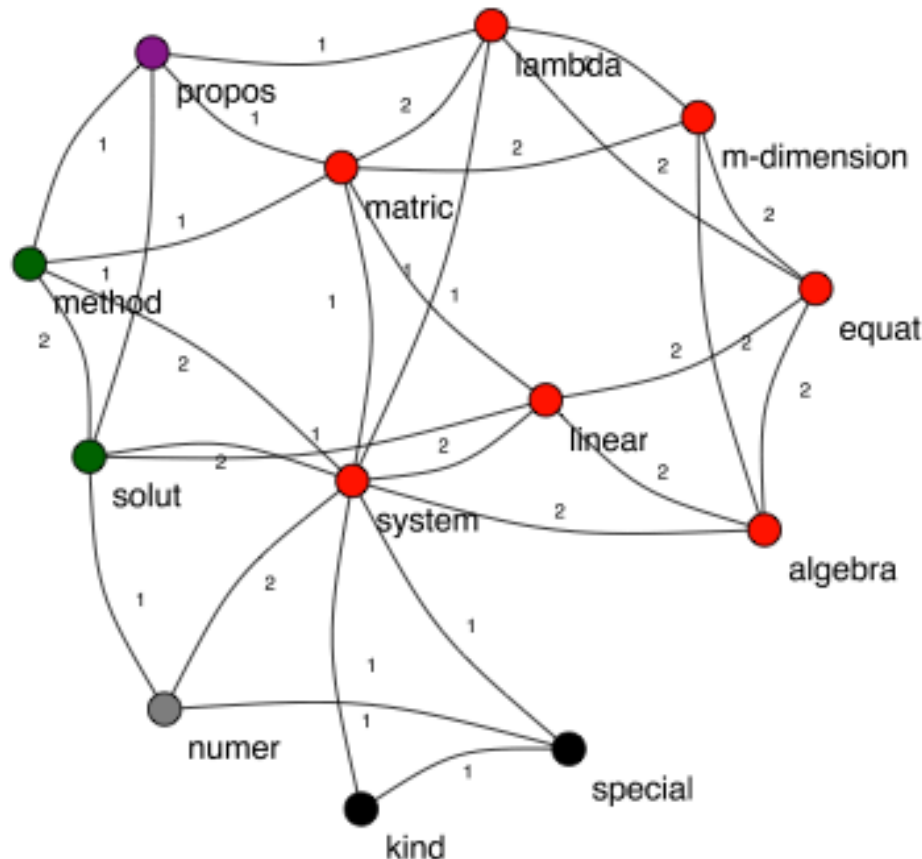
- Single-document keyword extraction
 - Select the most cohesive sets of words in the graph as keywords
 - Use k-core decomposition to extract the main core of the graph
 - Weighted edges

A method for solution of systems of linear algebraic equations with m-dimensional lambda matrices.
A system of linear algebraic equations with m-dimensional lambda matrices is considered. The proposed method of searching for the solution of this system lies in reducing it to a numerical system of a special kind.



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

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]

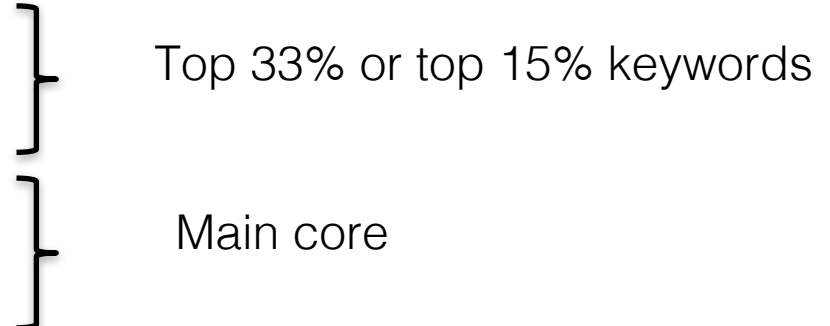
All approaches are **unsupervised** and **single-document**

Models and Baseline Methods

Graph-of-words:

- Undirected edges
- Forward edges
 - Natural flow of the text
 - An edge **term1** → **term2** meaning that **term1** precedes **term2** in a sliding window
- Backward edges

Keyword extractors:

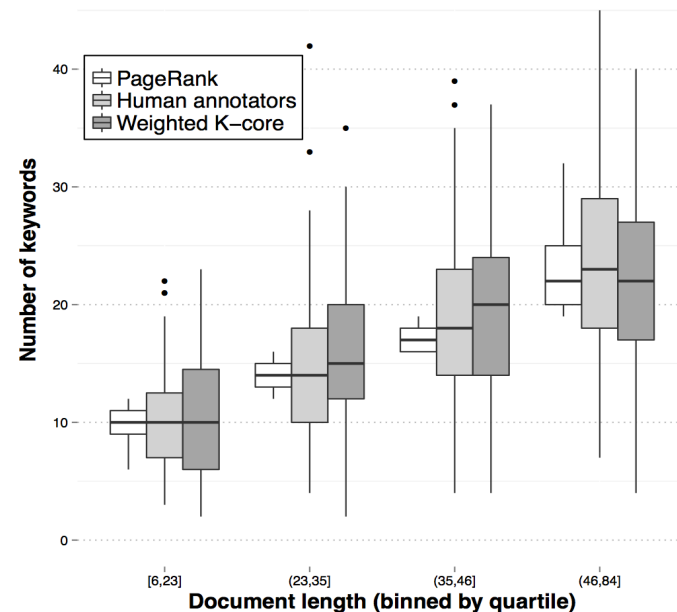
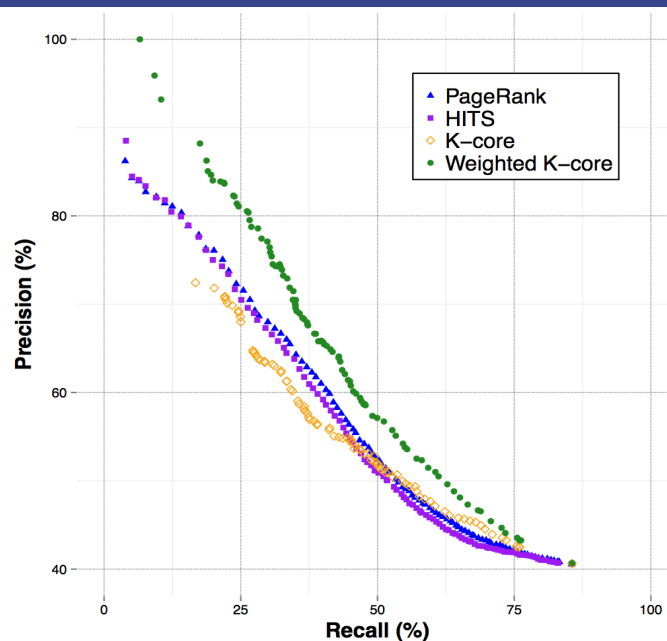
- PageRank
 - HITS (authority scores only)
 - k-core
 - Weighted k-core
- 
- Top 33% or top 15% keywords
- Main core

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

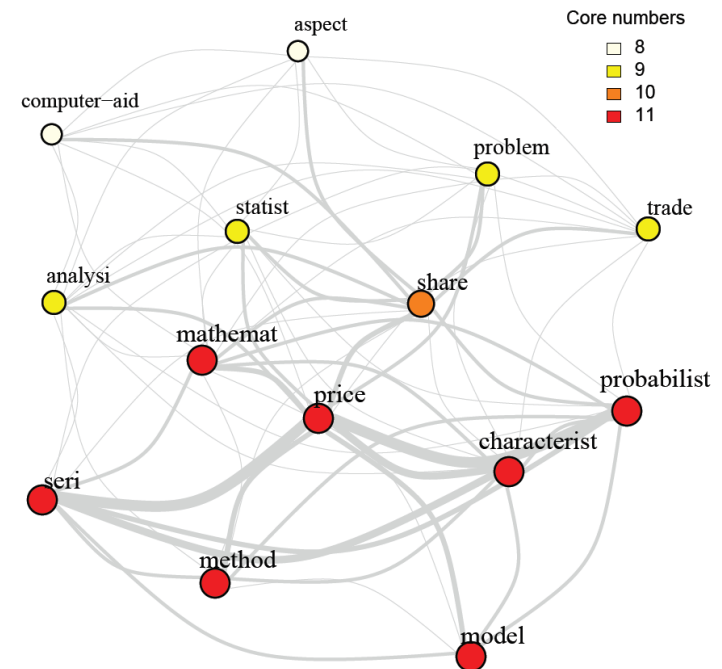
Example – ECIR'15 Paper

- 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

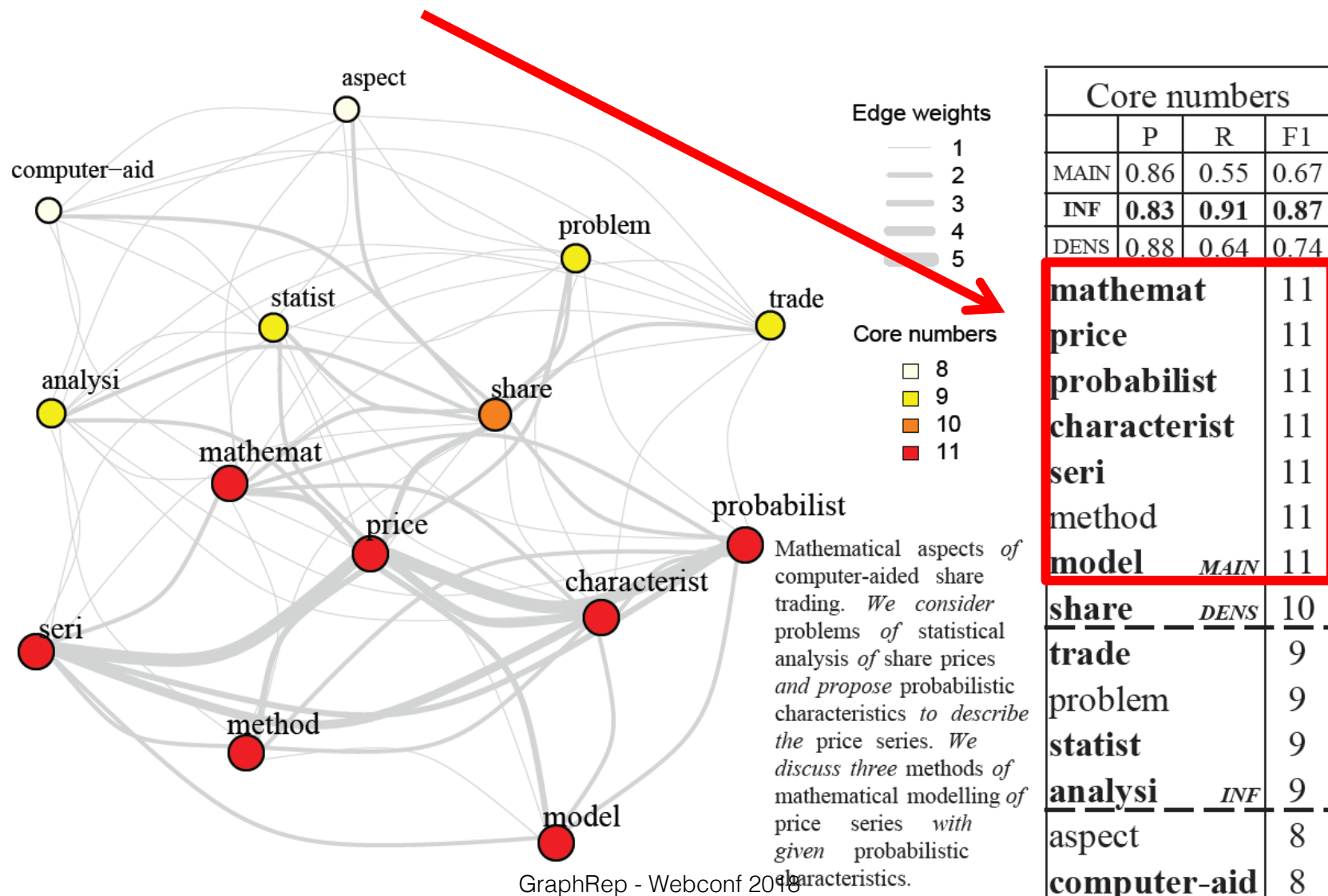
Graph degeneracy:

- 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)



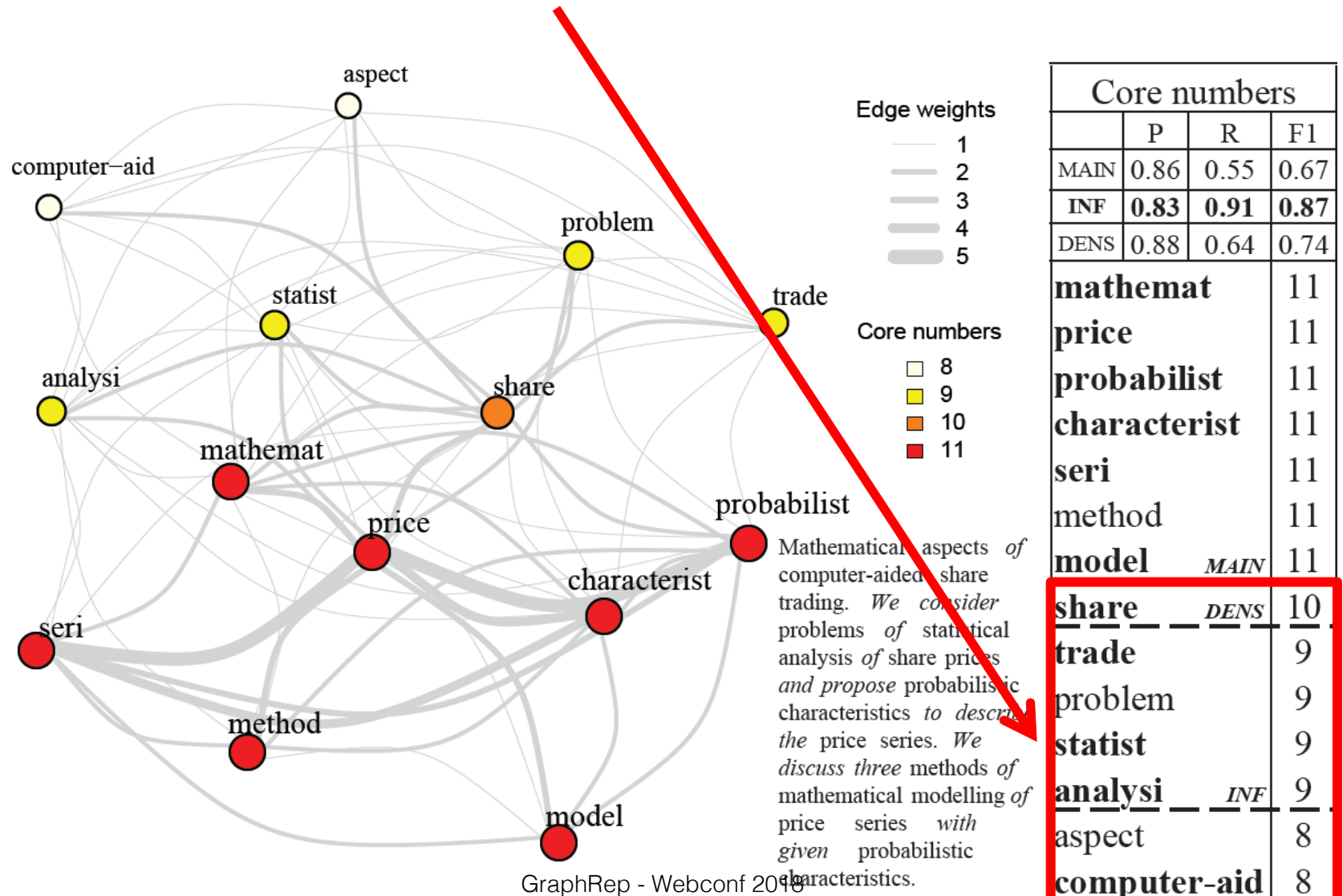
Drawbacks of Graph Degeneracy (1/4)

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



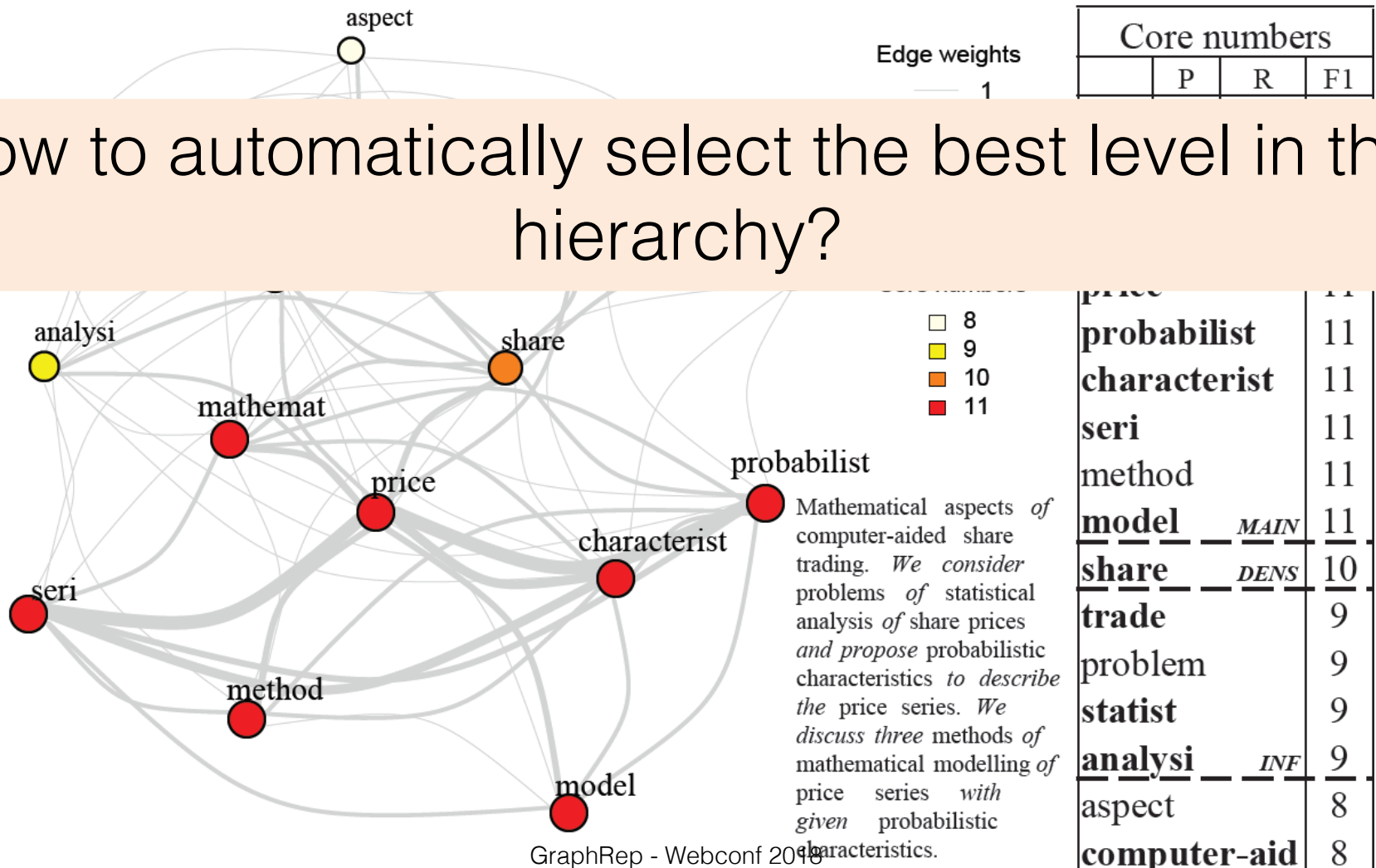
Drawbacks of Graph Degeneracy (2/4)

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



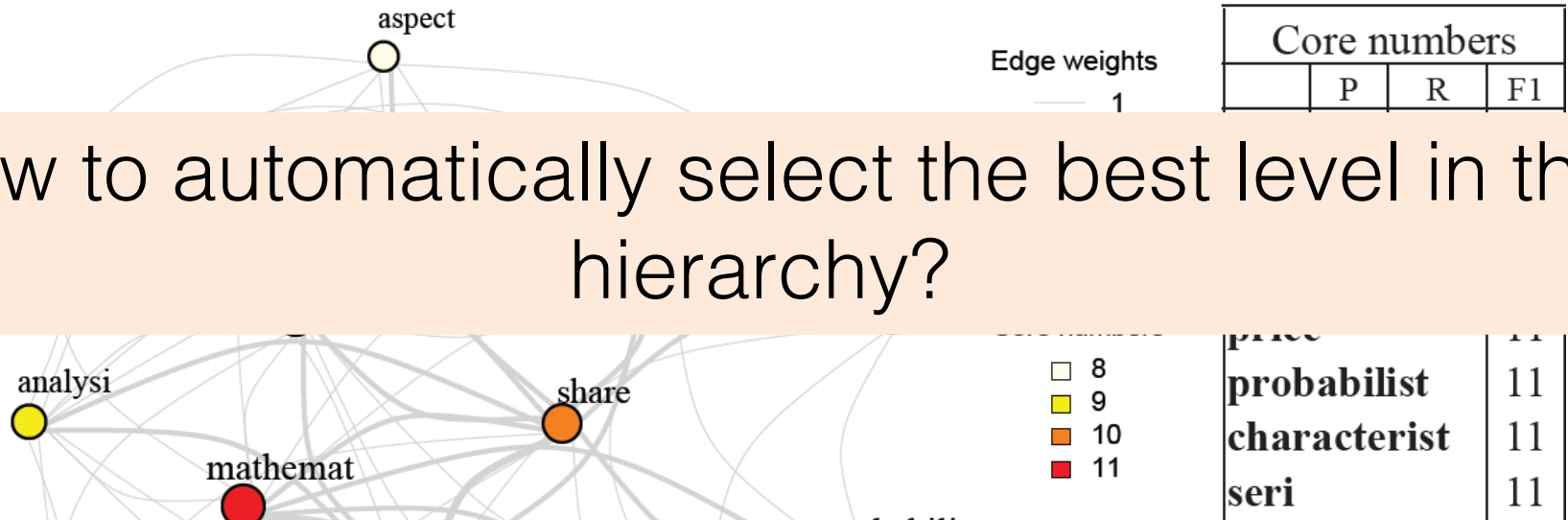
Drawbacks of Graph Degeneracy (3/4)

How to automatically select the best level in the hierarchy?

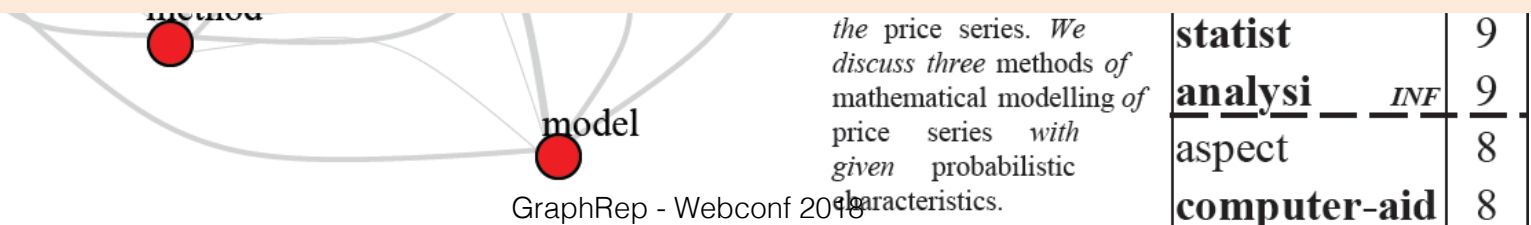


Drawbacks of Graph Degeneracy (4/4)

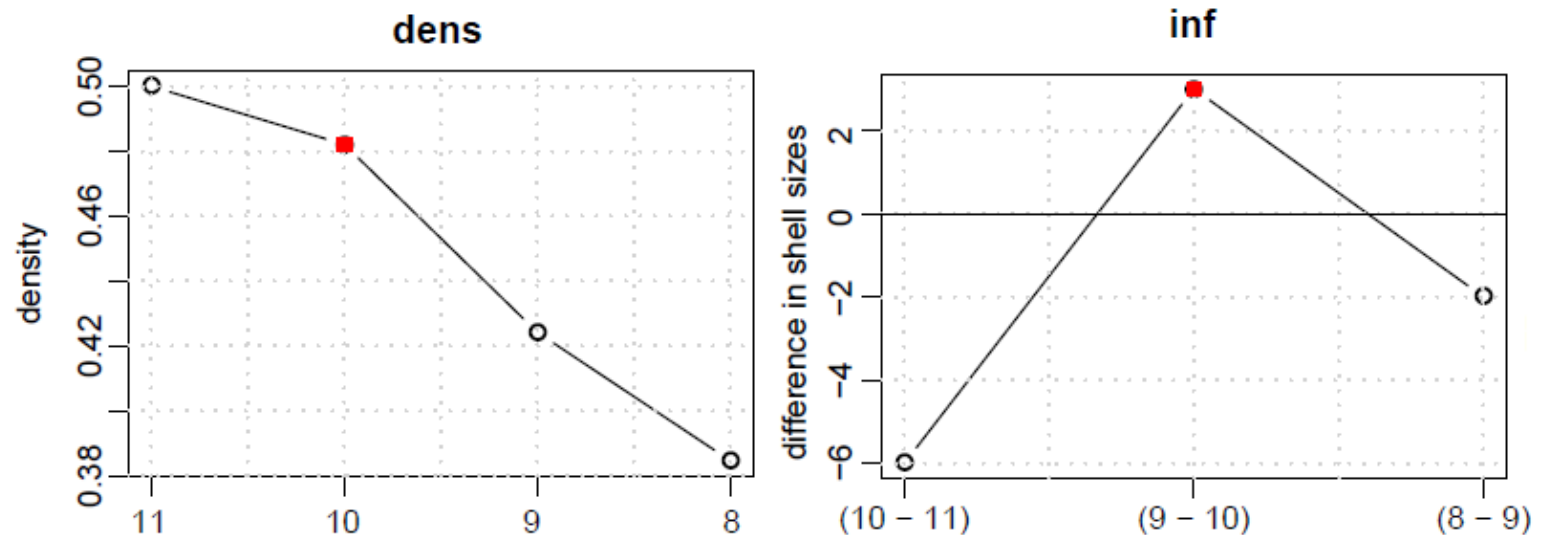
How to automatically select the best level in the hierarchy?



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



Graph Degeneracy for Keyword Extraction

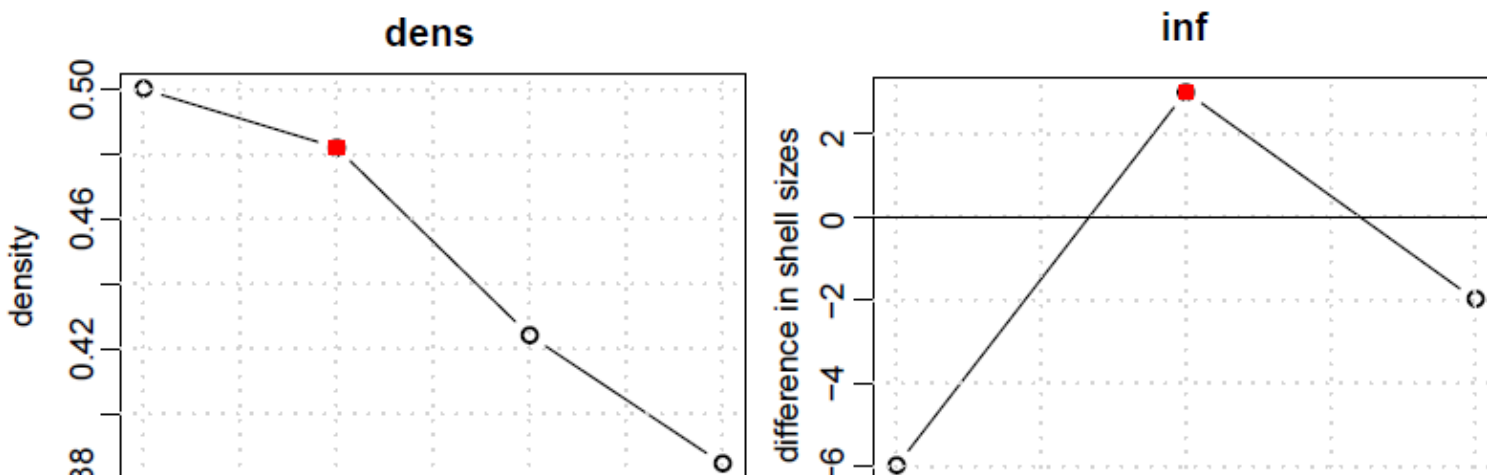


Heuristics:

- **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)

Graph Degeneracy for Keyword Extraction

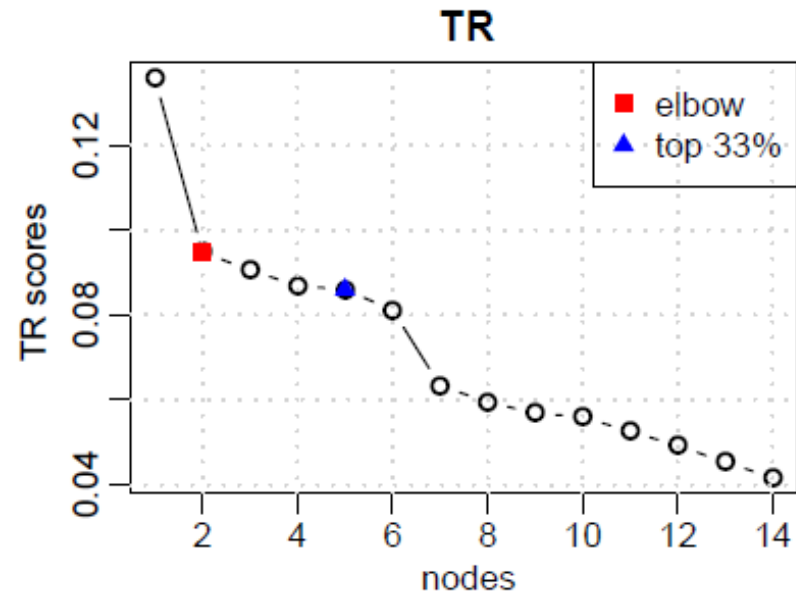
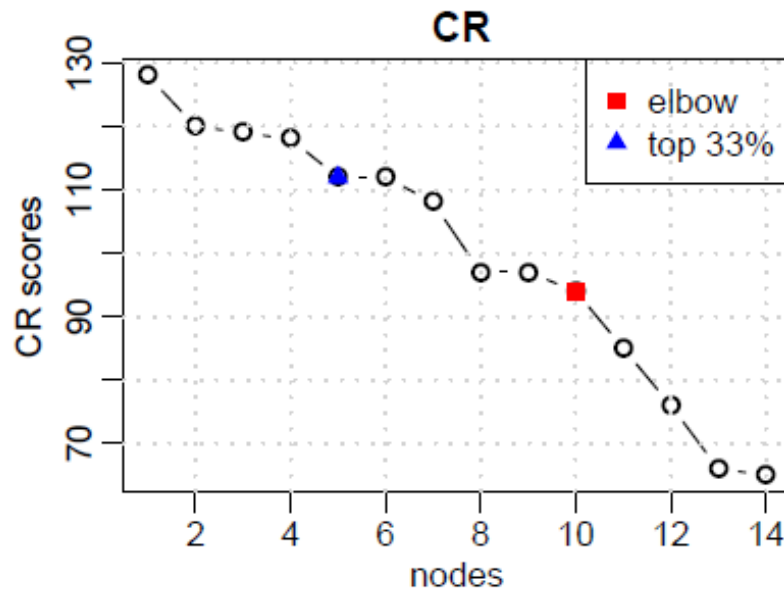


How to work at the node level while still retaining the valuable cohesiveness information captured by degeneracy?

- **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)

CoreRank

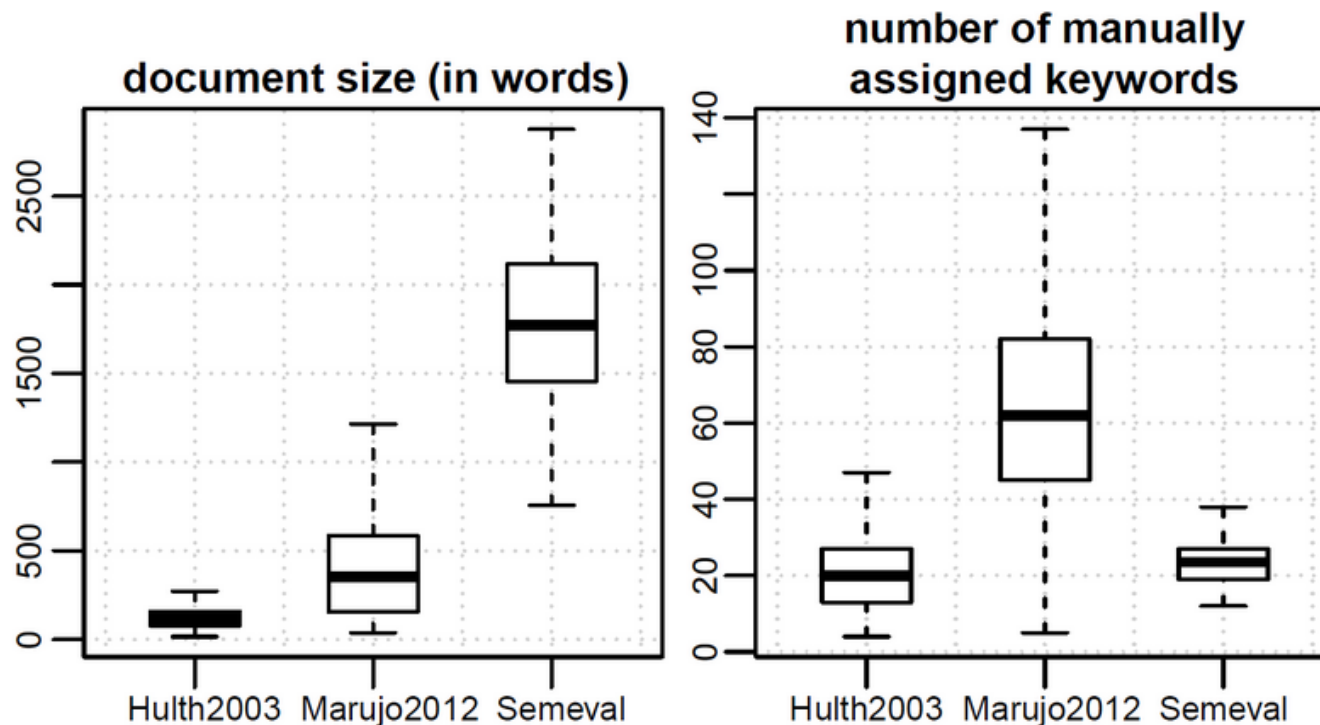


CoreRank (CR):

- 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

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		precision	recall	F1-score		precision	recall	F1-score
dens	48.79	72.78	56.09*	dens	47.62	71.46	52.94*	dens	8.44	79.45	15.06
inf	48.96	72.19	55.98*	inf	53.88	57.54	49.10*	inf	17.70	65.53	26.68
CRP	61.53	38.73	45.75	CRP	54.88	36.01	40.75	CRP	49.67	32.88	38.98*
CRE	65.33	37.90	44.11	CRE	63.17	25.77	34.41	CRE	25.82	58.80	34.86
main [†]	51.95	54.99	50.49	main [†]	64.05	34.02	36.44	main [†]	25.73	49.61	32.83
TRP [†]	65.43	41.37	48.79	TRP [†]	55.96	36.48	41.44	TRP [†]	47.93	31.74	37.64
TRE [†]	71.34	36.44	45.77	TRE [†]	65.50	21.32	30.68	TRE [†]	33.87	46.08	37.55

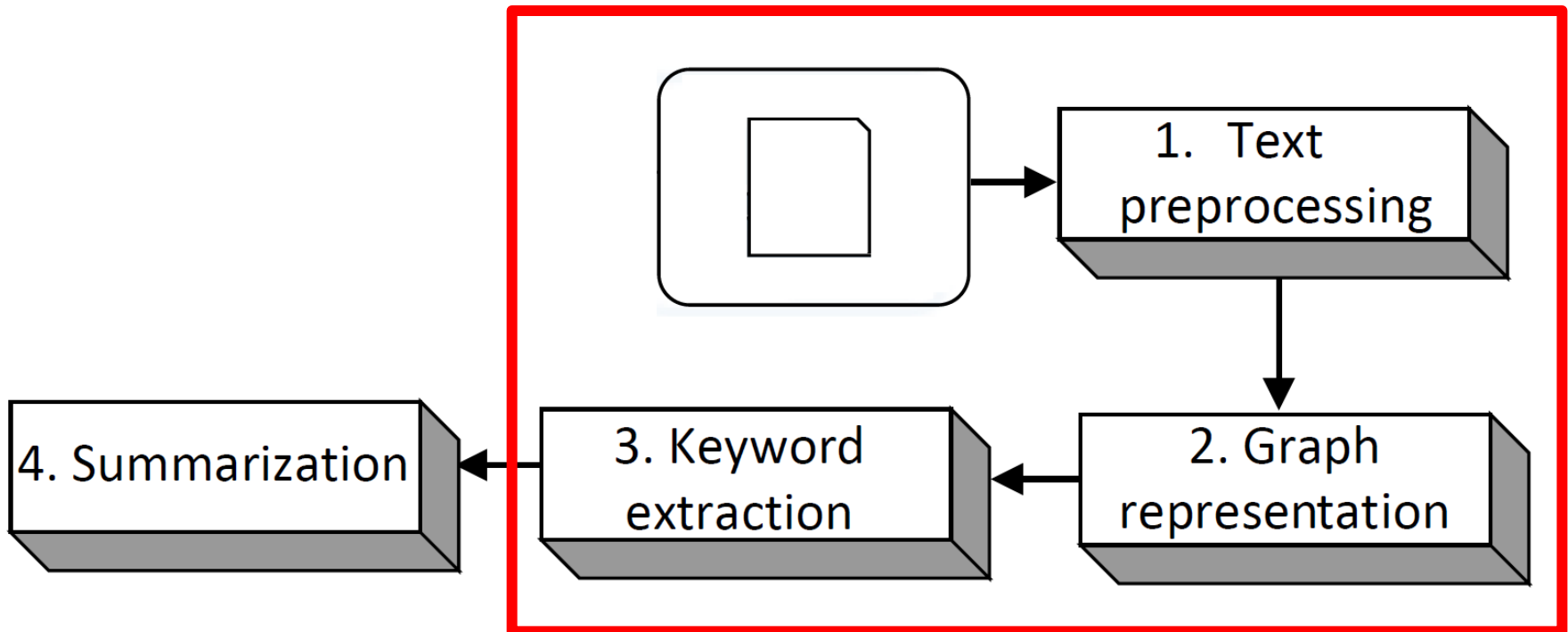
Hulth2003, K -truss, $W = 11$. *stat. sign. ($p < 0.001$) w.r.t. all baselines[†] Marujo2012, k -core, $W = 13$. *stat. sign. ($p < 0.001$) w.r.t. all baselines[†] Semeval, K -truss, $W = 20$. *stat. sign. ($p < 0.001$) w.r.t. *main*

- 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

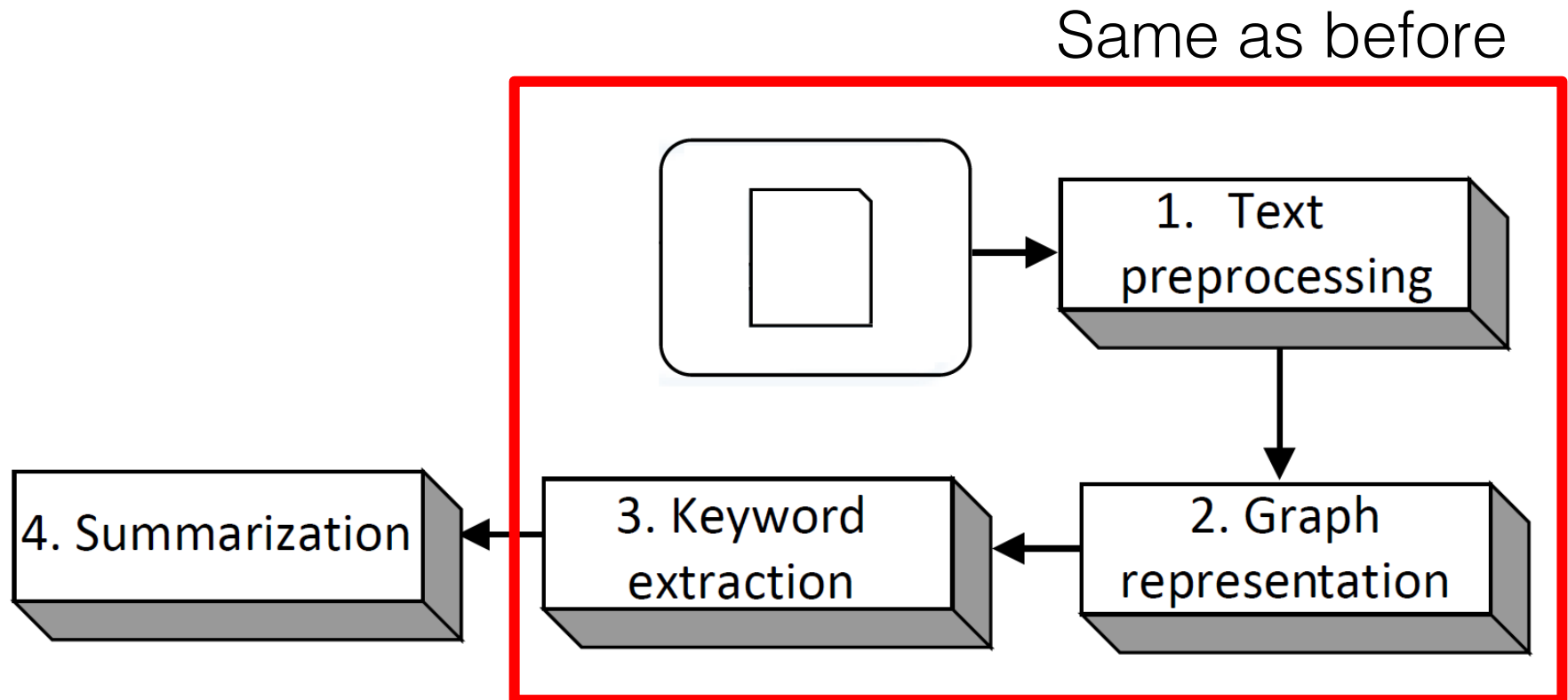
Extractive summarization

Extension to Extractive Document Summarization

Same as before



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)

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



$$F(G \cup v) - F(G)$$

scaled cost


$$c_v^r$$


- **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)

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

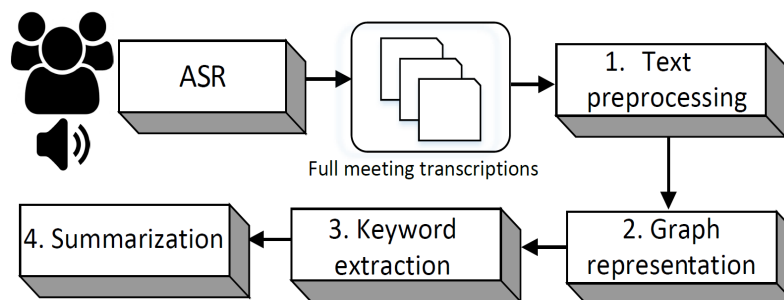

$$L(S) = \sum_{i \in S} n_i w_i \quad R(S) = N_{keywords \in S} / N_{keywords}$$

weighted sum of the keywords
contained in the summary

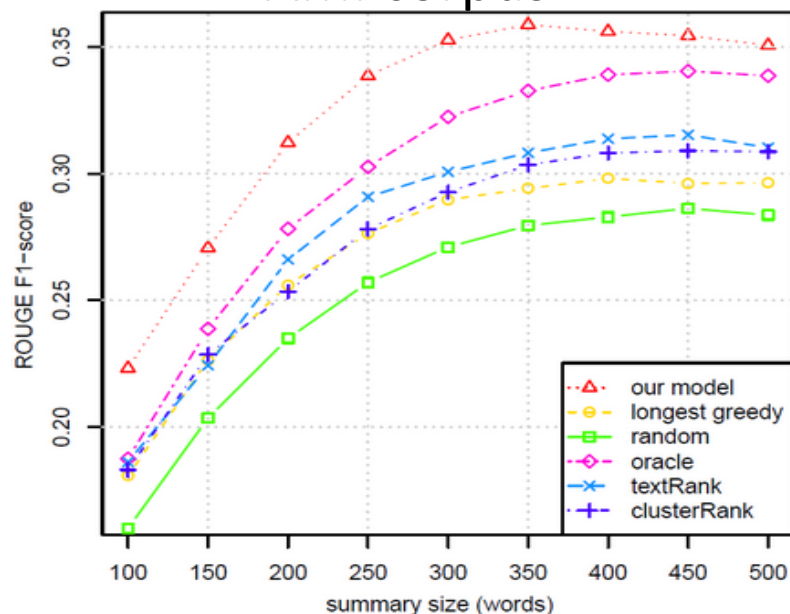
proportion of unique keywords
contained

Extractive Document Summarization (4/4)

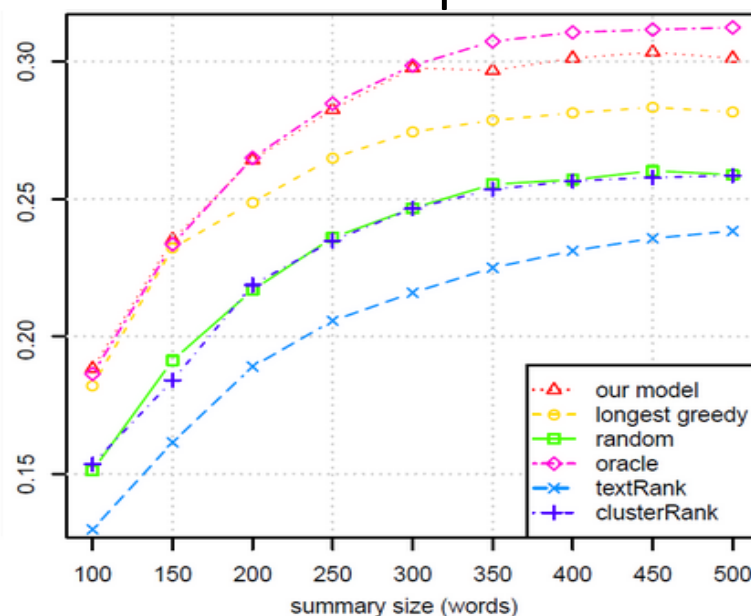
Tested for multiparty virtual meetings summarization:



AMI corpus



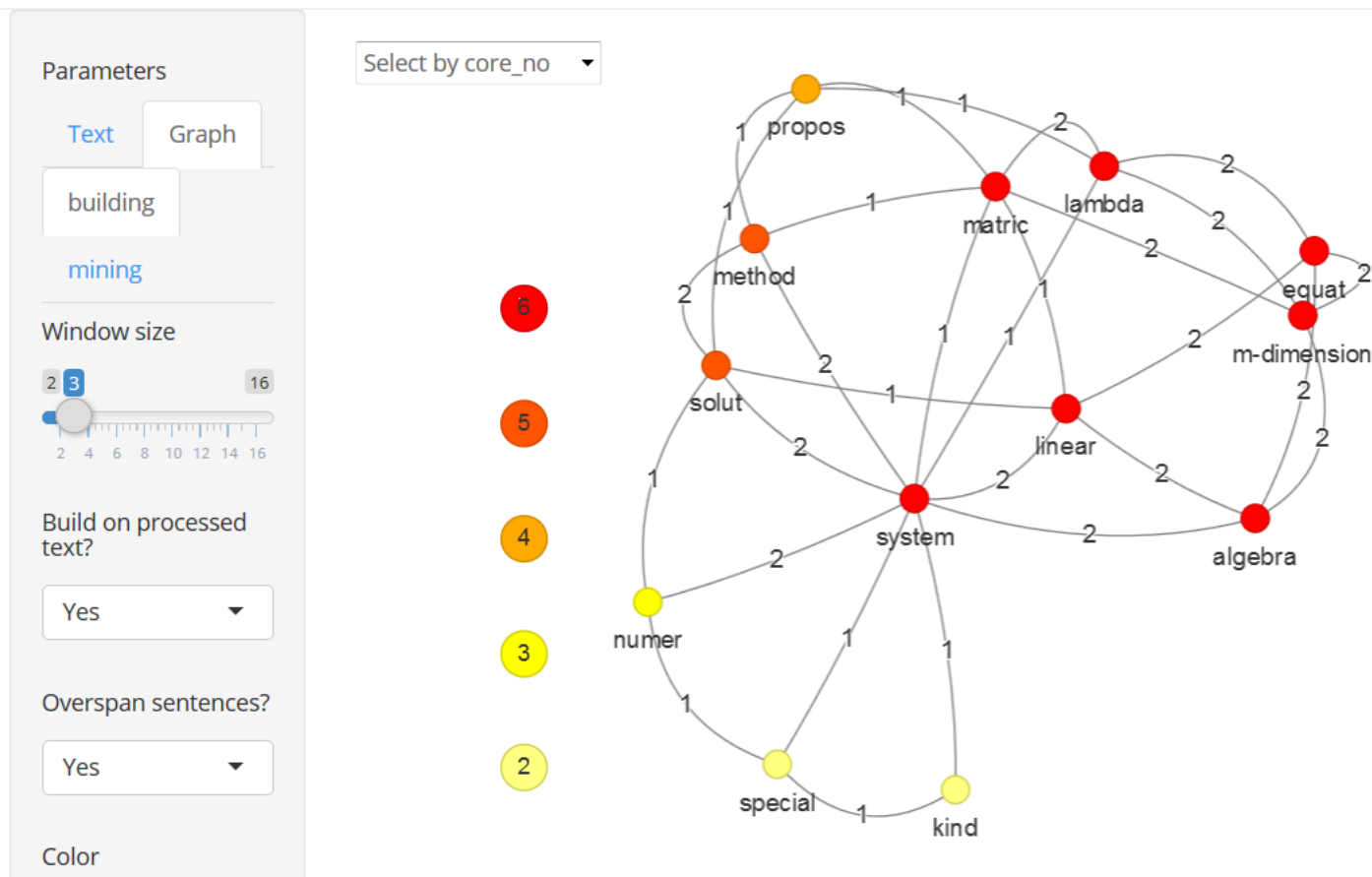
ICSI corpus



GoWvis visualization tool

GoWvis Visualization Tool

A method for solution of systems of linear algebraic equations with m -dimensional lambda matrices. A system of linear algebraic equations with m -dimensional lambda matrices is considered. The proposed method of searching for the solution of this system lies in reducing it to a numerical system of linear algebraic equations.



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

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/>

Abstractive Summarization

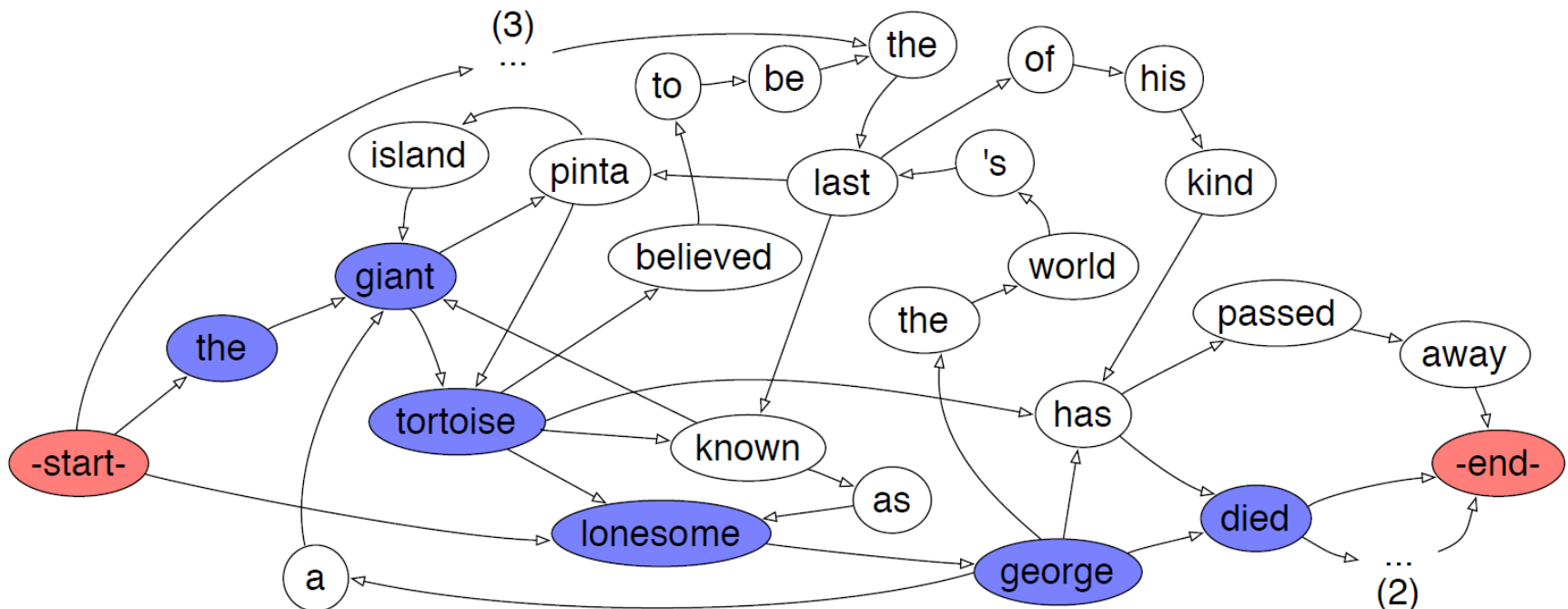
Multi-sentence compression in word graphs

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

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):

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

Where:

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

- **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):

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

Where:

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

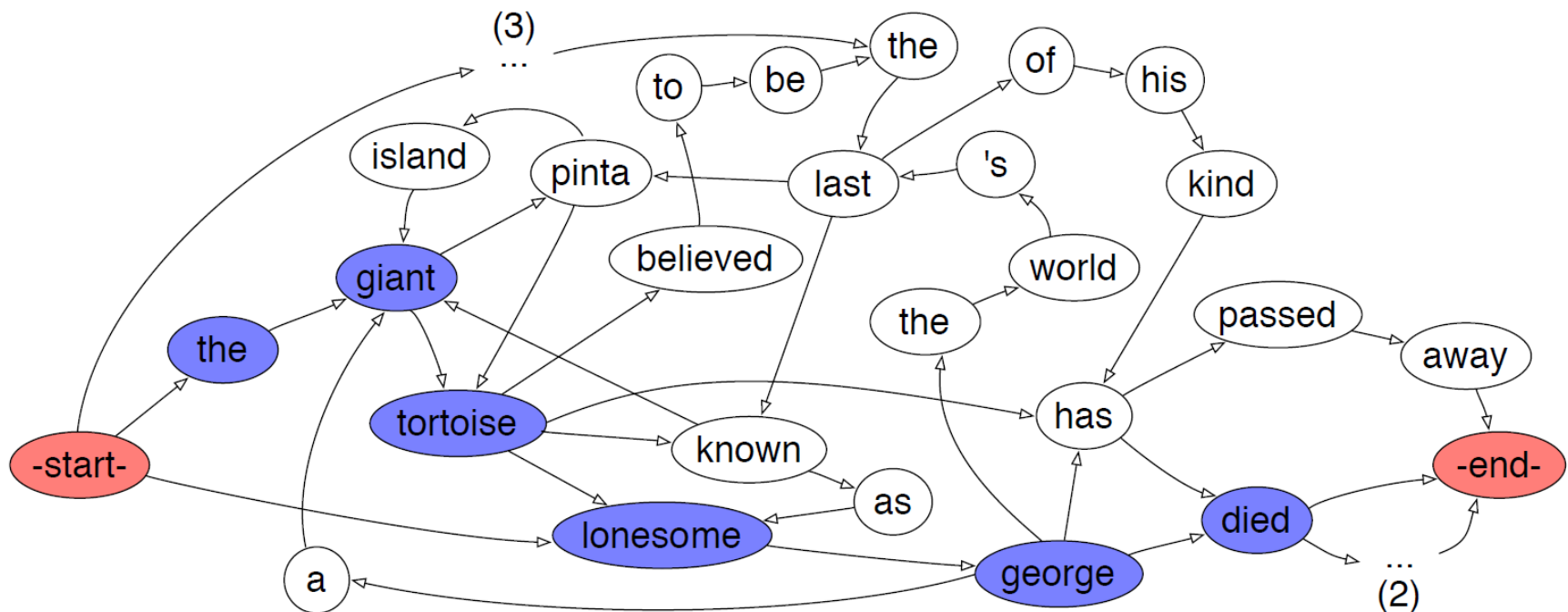
- **freq(i)** is the number of words that have been mapped to node **i**
- **diff(s,i,j)** is the distance between word **i** and word **j** in sentence **s**
- Intuition for (1):
 - Eq. (2) is a measure of cohesion between 2 words, but disregards the individual importance of the words => we need to 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

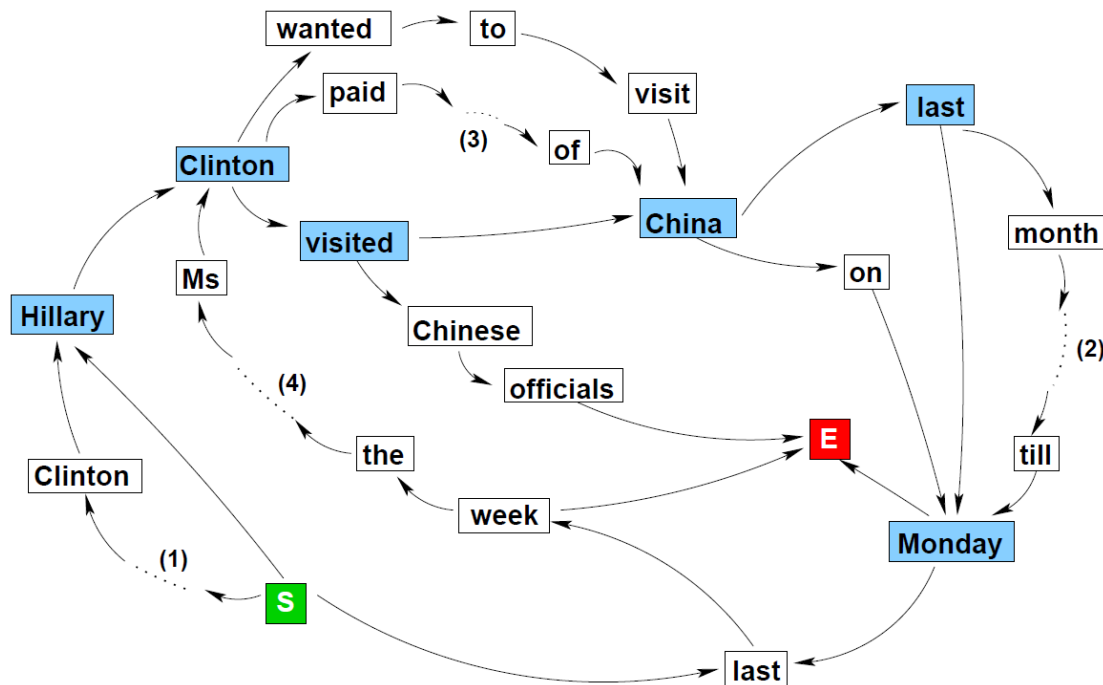
Examples

- 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

Event detection in text streams

1: Threshold based approach

[AAAI – ICWSM 2015]

Introduction - Twitter

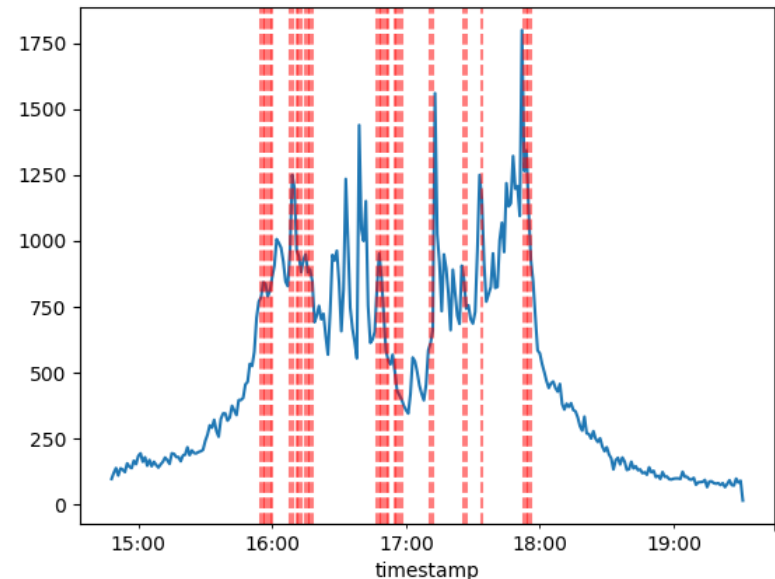
- Very popular microblogging service
 - 330 million active users
- Real time messages known as tweets
 - Instantaneous nature
 - Major communication medium
- Why Twitter
 - Large variety of usages: communication, news, politics, celebrities
 - Huge volume of data
 - Transmission in real time

Introduction - Event Detection

- Users report latest news or comment about real-world events
 - Events consisting of a sequence of important moments
 - Events not covered systematically by traditional media
 - Interest in tracking the evolution of an event
- Challenges:
 - Huge volume of tweets - Noisy content
 - Heterogeneous users
 - Multiple hashtags per event
 - Generating short summary

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?

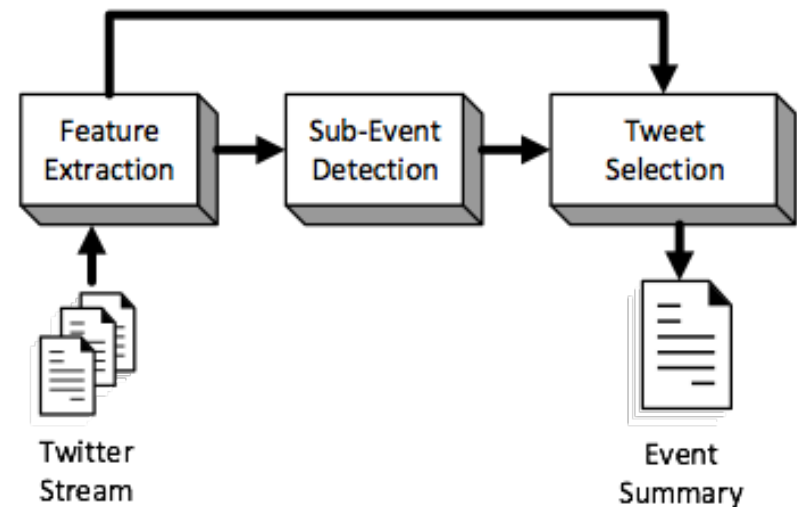


- Tweet Rate of a football Match
- Red lines: True Events
- Very noisy during the event!

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
1. **Sub-event detection:** decides whether a sub-event has occurred
1. **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

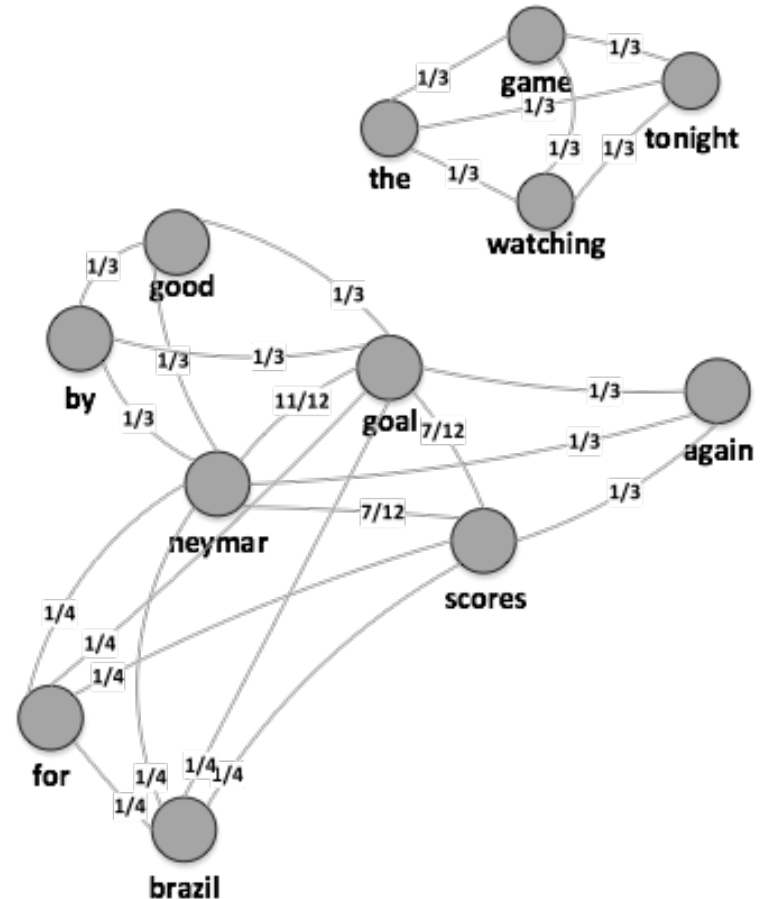
Graph-based Representation of Tweets

- 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

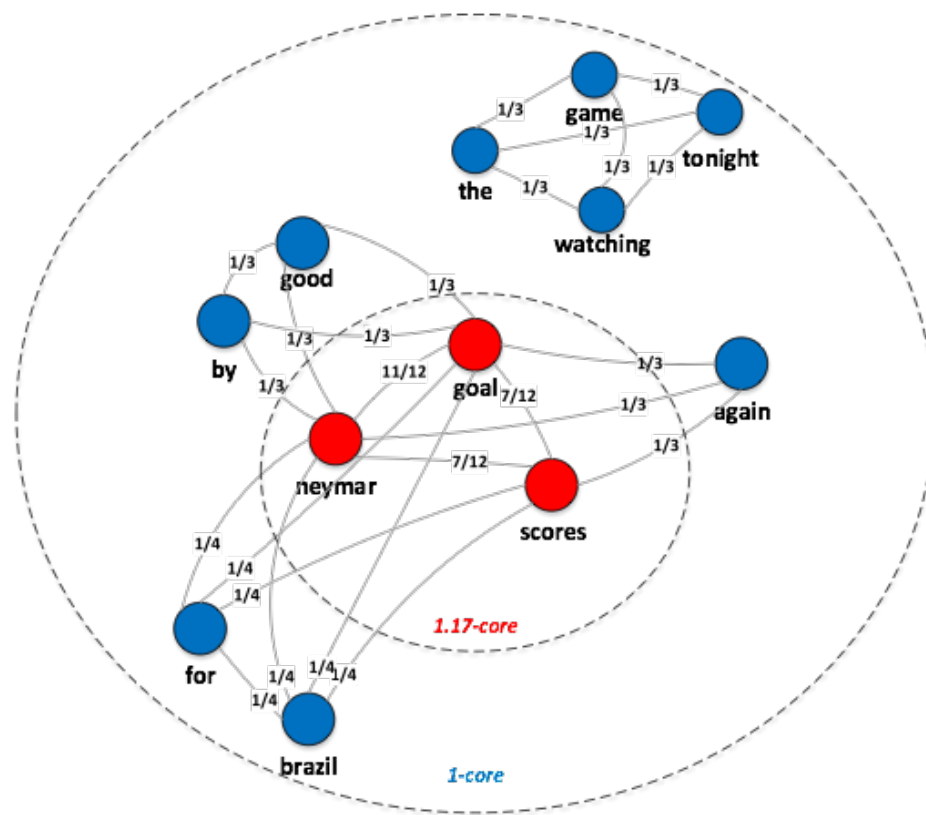
Dataset: tweets from the 2014 FIFA World Cup in Brazil



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

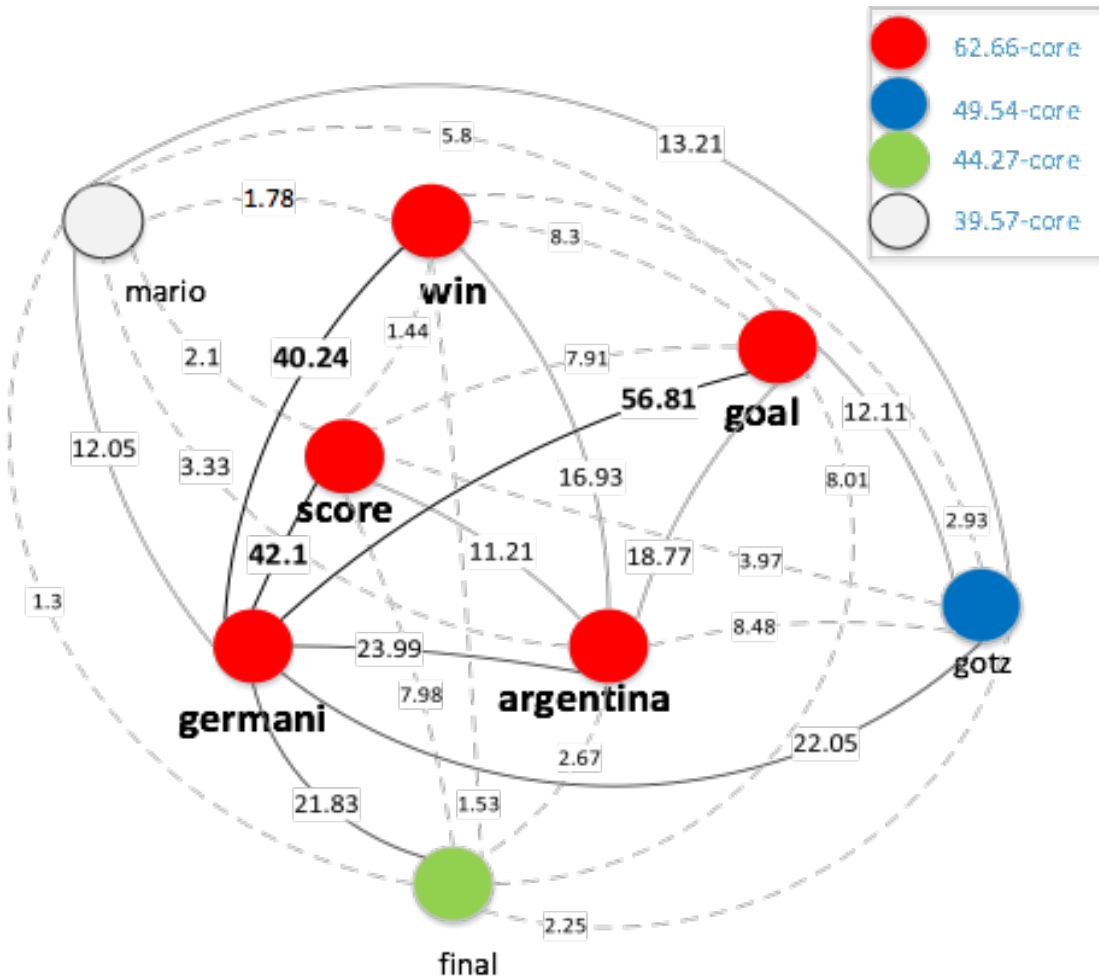
$$\sum_{i=1}^d c_i^t > \theta \times \frac{1}{p} \sum_{j=t-p}^{t-1} \sum_{i=1}^d c_i^j$$

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

Experimental Setup



Baselines-Approaches

(Detection -Term Weight)

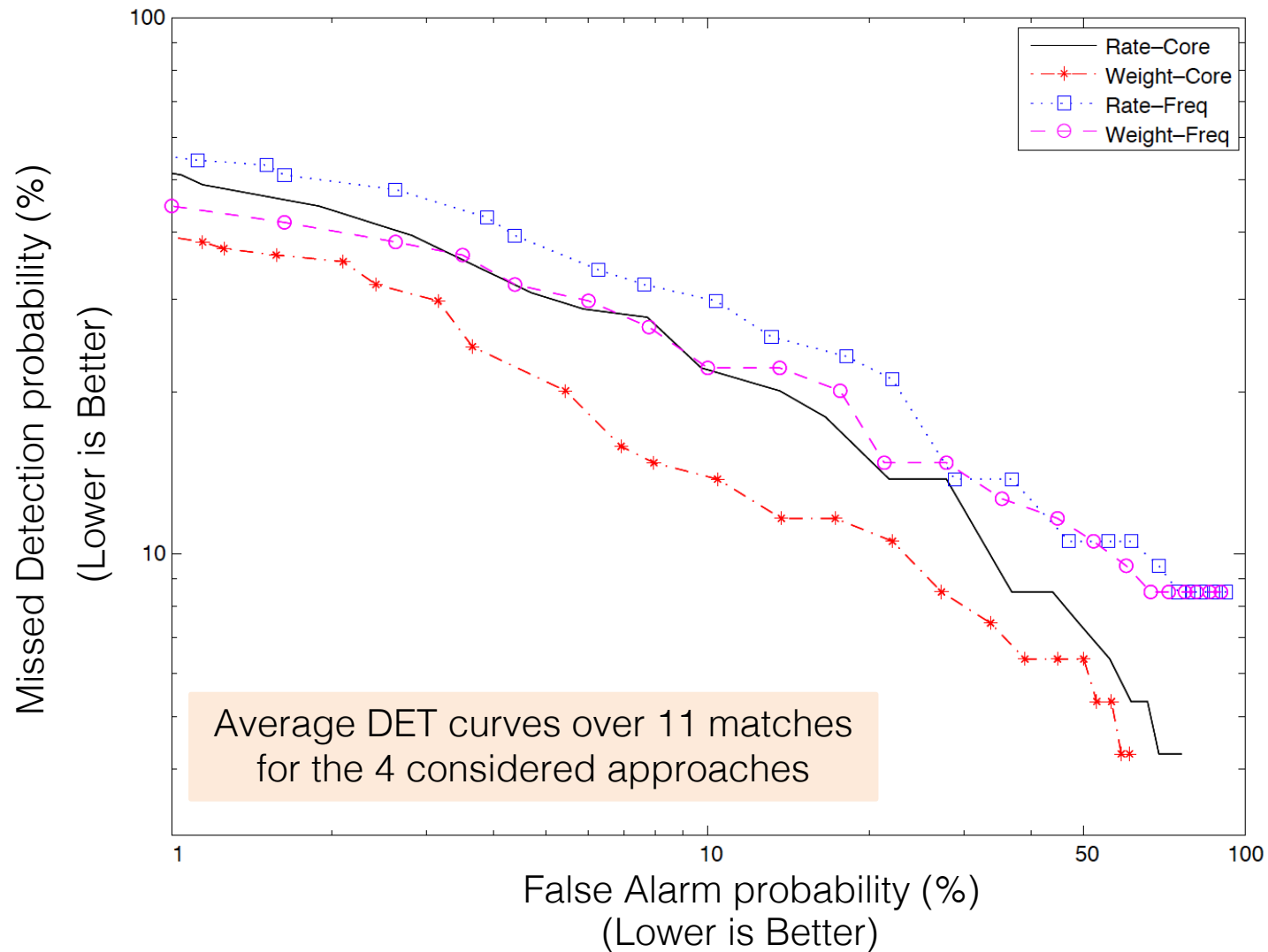
- Rate-Freq: the common baseline
- Rate-Core
- Weight-Core: Our approach
- Weight-Freq

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.

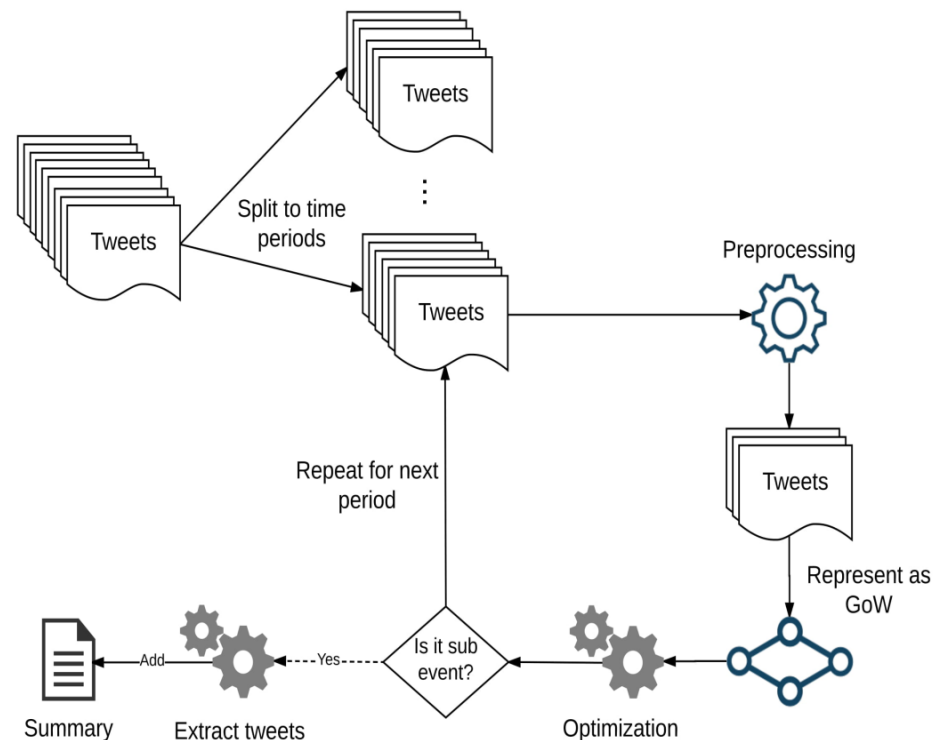
Summary of the Argentina vs. Belgium match generated automatically using Weight-Core and manually by ESPN

Event detection in text streams

2: An Optimization Approach for Sub-event Detection and Summarization in Twitter [ECIR 2018]

Introduction - System Architecture

- Tweets decomposition into time intervals
- Graph of Words representation
- Convex optimization formulation
- Summarization



Methodology - Data Preprocessing

Given a set of raw tweets:

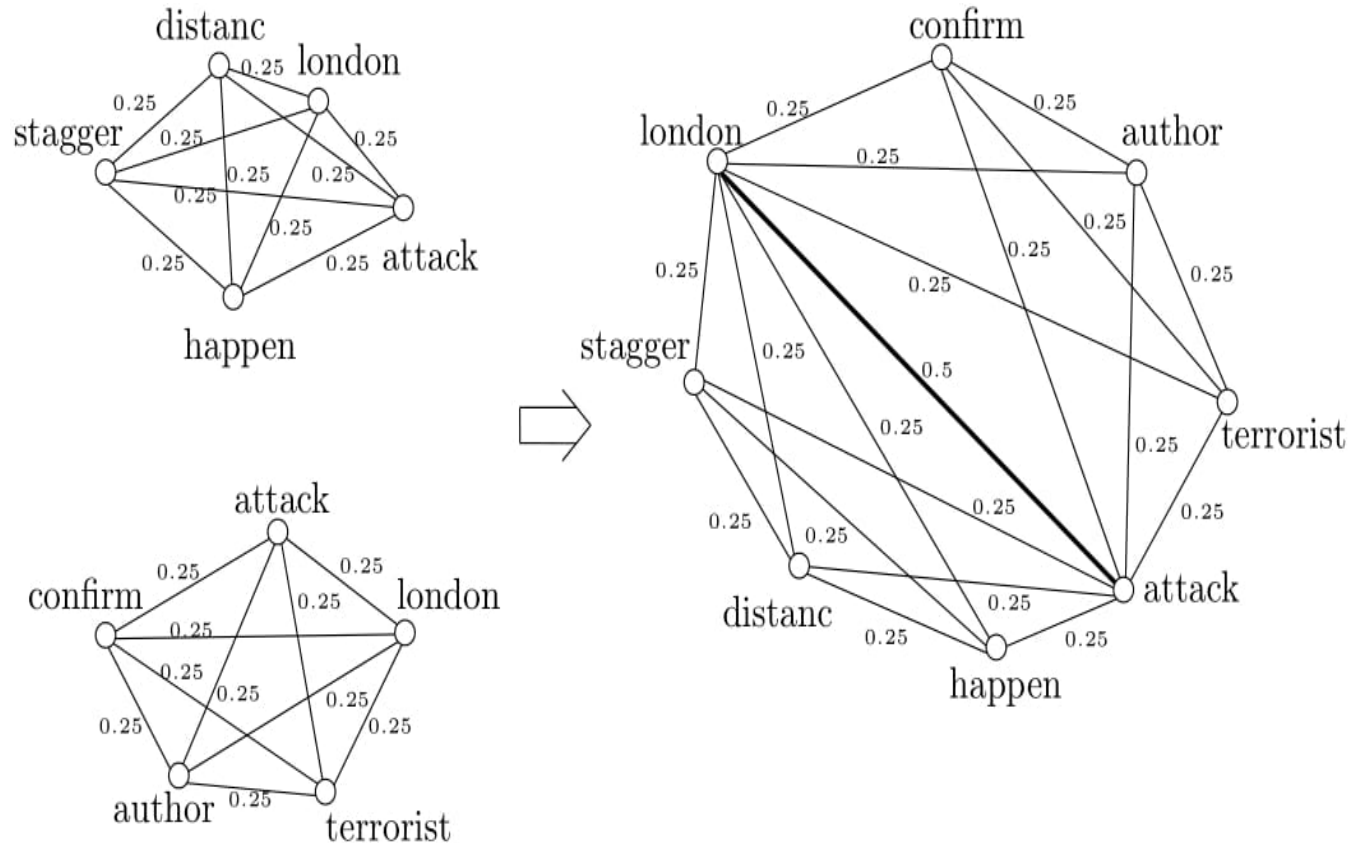
- Remove not relevant tweets
 - Retweets
 - Duplicates
 - Mentions
- Text preprocessing:
 - Tokenization
 - Stopword removal
 - Punctuation, special character and URL removal
 - Stemming using Porter's algorithm

Methodology - Graph of Words

Given a set of pre-processed tweets:

- Each tweet represented as a weighted graph
 - Vertices correspond to unique terms
 - Edges are drawn between all pairs of vertices (clique)
 - Weight set equal to $1/(n-1)$ n : number of unique terms in the tweet
- Creating a graph G_i corresponding to the time period i
 - Add tweet-graphs sequentially
 - Increase weight of edges occurring in multiple tweets

Methodology - Graph of Words



Methodology - Sub-Event Detection

Sub-Event detection:

- Change rate among tweets posted currently compared to previous time periods
- Pairs of words with high co-occurrence frequency.

- Detection handled as *optimization problem*

- For time period i , graph G_i
- Adjacency Matrix $\mathbf{A}_i \in \mathbb{R}^{n \times n}$
- Represents co-occurrence among words in tweets

$$\mathbf{A}_i = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_n \\ | & | & & | \end{bmatrix}$$

- b is the n^2 rows matrix representing the co-occurrence of words in the tweets in the periods p

$$\mathbf{b} = \text{vec}(\mathbf{A}_i) = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}$$

Methodology - Sub-Event Detection

- Construct matrix \mathbf{W} with weights from previous p periods
- each \mathbf{w}_i represents co-occurrence weights of all terms in the graph time i (summarizing previous tweets)

$$\mathbf{W} = \left[\begin{array}{c|c|c} & & \\ \mathbf{w}_{i-p} & \dots & \mathbf{w}_{i-1} \\ & & \end{array} \right] \quad \mathbf{W} \in \mathbb{R}^{n^2 \times p}$$

- Optimization problem formulated as:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2} \|\mathbf{W}\mathbf{x} - \mathbf{b}\|_2^2 \\ \text{s.t.} \quad & \mathbf{1}^\top \mathbf{x} = 1 \\ & x_i \geq 0, \quad \forall i = 1, \dots, p \end{aligned}$$

- \mathbf{x} represents vector matching best the current to the previous p times

Methodology - Sub-Event Detection

- Let $\mathbf{c} = \mathbf{W}\mathbf{x}^*$
- then $\frac{1}{2}||\mathbf{c} - \mathbf{b}||_2^2$ represents the shift based on the last period word concurrences
- Large values signify events

$$\frac{1}{2}||\mathbf{c} - \mathbf{b}||_2^2 \geq \theta$$

- θ is a parameter to be learned (further on)

Methodology - Summarization (1/2)

- Extractive summarization algorithm → Select a subset of tweets serving as summary of the sub-event
- **Goal:** Selected tweets containing most significant terms of the detected sub-event
- Define a monotone sub modular function f that rewards both the **coverage** and the **diversity** of the set
- proposed function f can identify multiple sub-events in a single time period

Methodology - Summarization (2/2)

A sub-event is best described by tweets whose graph-of-words contains multiple “important” edges

- G_i graph-of-words representation of tweets posted during time period i
- $\mathcal{S} \subseteq D_i$ a set of tweets extracted from time period i
- $f(\mathcal{S})$ the sum of weights of the edges of G_i that are “covered” by tweets belonging to \mathcal{S}
- f monotone non-decreasing and submodular \rightarrow use greedy approximation algorithm
- guaranteed to be within $(1 - 1/e) \approx 0.63$ of the optimal solution

Evaluation - Experiments

Baseline method

- Tweeting rate (Burst)
 - Number of tweets in period
 - Ignores previous periods

Evaluation settings:

- Threshold: Training process separate for each match
- Period duration: 1 min

Evaluation - Dataset

- 18 FIFA World Cup 2014 and 2 FIFA World Cup 2010 matches
 - ❑ 3 used for tuning and the rest for testing
- Only some sub-event types considered
 - ❑ goals, penalties, cards, match start, match end and half time
 - ❑ injuries, substitutions, free kicks etc. ignored
- Ground truth
 - ❑ Manual annotation of events

Match	Actual Events	Collected Tweets	Preprocessed Tweets
ARG - BEL	7	313,803	108,250
ARG - GER	9	824,241	262,112
AUS - NED	12	96,834	25,997
AUS - ESP	9	86,843	13,608
BEL - KOR	7	99,192	32,053
CMR - BRA	11	148,298	35,085
FRA - GER	6	525,725	160,727
FRA - NGA	6	367,899	128,718
GER - ALG	8	712,525	276,227
GER - BRA	12	973,985	295,875
GER - GHA	8	285,804	77,449
GER - USA	7	256,445	86,040
GRE - CIV	10	113,402	51,101
HON - SUI	8	41,539	10,082
MEX - CRO	11	155,549	36,981
NED - CHI	8	95,108	25,819
NED - MEX	10	628,698	217,472
POR - GHA	10	272,389	91,110
GER - SRB	14	45,024	29,062
USA - SVN	12	85,675	53,292
Total	185	6,128,978	2,017,060

Parameter Tuning

$$\frac{1}{2} \|\mathbf{c} - \mathbf{b}\|_2^2 \geq \theta$$

- Relationship between the optimal value of threshold θ and number of tweets – as a regression problem
- selected 3 matches as training examples
 - Rest 17 matches test set
- Exhaustive grid search, $\theta \in [1, 100]$
- Minimize the least-squares-error (x number of events)

$$\hat{\theta} = w_1 x^2 + w_2 x + w_3$$

Evaluation - Results

Metric Method	Macro-average			Micro-average		
	precision	recall	f1-score	precision	recall	f1-score
OptSumm	0.76	0.75	0.75	0.73	0.74	0.73
Burst	0.78	0.54	0.64	0.72	0.54	0.62

- Evaluate the proposed system (OptSumm) on the task of sub-event detection
- Standard measures in information retrieval
 - 1) Precision
 - 2) Recall
 - 3) F1-score
- OptSumm detected sub-events that could not be detected by Burst,

Evaluation - Sub-events Detected

- Key sub-event types, actual numbers vs. detected ones for 17 matches - test set
- Successfully detected all goals, own goals, penalties, red cards, match ends and half times, and almost all match starts
- failed to detect the yellow cards consistently

Event type	# actual events	# detected events
Goal	42	42
Own Goal	2	2
Penalty	3	3
Red Card	3	3
Yellow Card	51	15
Match Start	17	14
Match End	17	17
Half Time	17	17

Evaluation - Summary

- System produces informative and reasonable textual descriptions of the important moments.

Our Summary	FIFA
Underdog Nigeria vs. European giants France. Going to be a great match!	The match kicks off.
Nigeria awarded a free kick in a good position after Matuidi collides with Odemwingie. Nigeria looking decent on the break so far. #fra #nga	Matuidi (France) concedes a free-kick following a challenge on Odemwingie (Nigeria).
France #fra 0-0 Nigeria #nig - Nigeria with Eminike score, but ruled out for offside, good decision 18mins	Emenike (Nigeria) is adjudged to be in an offside position.
Pogbaaaaaa!!! excellent skill! made that entire move and ended it with a superb volley but keeper made a good save	Pogba (France) sees his effort hit the target.
Half time: France 0-0 Nigeria. Goalless in braşilia. tight game.	The referee brings the first half to an end.
54: Blaise Matuidi gets the first yellow card of the game after a nasty challenge.	Matuidi (France) is booked by the referee.
Nigeria the best team by far. That usually means France will scrape a lucky win.	-
Omg! #Benzema so close to scoring, just 2? inches short. Still 0-0 (Nigeria-France) in a suddenly very exciting match!	Benzema (France) sees his effort hit the target.
If the French don't score in this game, it would be a miracle for Nigeria. France has been inches away from about 3 goals at this point.	-
Goal France! Who else, but the future, Paul Pogba, heading into an open net. Finally les blues score. 1-0, 80th min.	Pogba (France) scores!!
Gooo! France scores in the 91st minute! Partial score, France 2-0 Nigeria. #worldcup goal count - 147.	Yobo (Nigeria) scores an own goal!!
Full-time: #fra 2-0 #nga. France book their spot in the quarter-final while Nigeria crash out of the 2014 Fifa world cup	The final whistle sounds.

Evaluation - Other events

- Westminster bridge (London)
 - Terrorist attack on 22 March 2017
 - 5 people died, at least 50 were injured
- Collected 153, 485 tweets
 - Time period = 15 minutes
 - Taking into consideration only the previous period



Evaluation - Other events

Timestamp	Description
22 Mar 14:56	#BREAKING London Terror Attack - Parliament lockdown, man with knife shot by security. Many people ran down by car on Westminster Bridge. ◇ REPORTS AT LEAST 12 INJURED IN A CAR RAMMING/STABBING ATTACK OUTSIDE PARLIAMENT IN LONDON, UK. ASSAILANT SHOT BY POLICE.
22 Mar 15:26	#DEVELOPING: London police are treating the attack at Westminster near British Parliament "as a terrorist incident until we know otherwise." ◇ #BREAKING Met confirm terror attack. Officer stabbed, shots heard at Parliament London. Many injured on Westminster Bridge hit by vehicle.
22 Mar 17:56	4 people dead, including police officer, and at least 20 injured in Parliament attack #Parliament London#terrorism. ◇ Thoughts with the victims of today's terrorist attack, their families & all those caught up at #Westminster. London will #KeepCalmAndCarryOn.
22 Mar 22:26	Metropolitan Police say 5 people have died and around 40 have been injured in the Westminster terror attack in London today #londonattack. ◇ 4 people including a terrorist and a police officer have been killed and 20 others injured in an terror attack in London.
23 Mar 10:26	Seven arrested as police raid addresses in Birmingham and London linked to Westminster terror attack #York. ◇ Mashable "Donald Trump Jr. tweeted about the London attack and got instantly dragged" #News.
23 Mar 11:56	BREAKING: Islamic State claims responsibility for terror attack outside Parliament in London #Westminster #londonattack. ◇ News agency linked to ISIS is out with a claim of responsibility from the terrorist organization for yesterday's deadly attack in #London.
23 Mar 15:26	Breaking: London police identify man behind UK terror attack as 52-year Khalid Masood, a Briton with a criminal history. ◇ London attack: Westminster terrorist named as Khalid Masood by Scotland Yard after Isil claims responsibility.

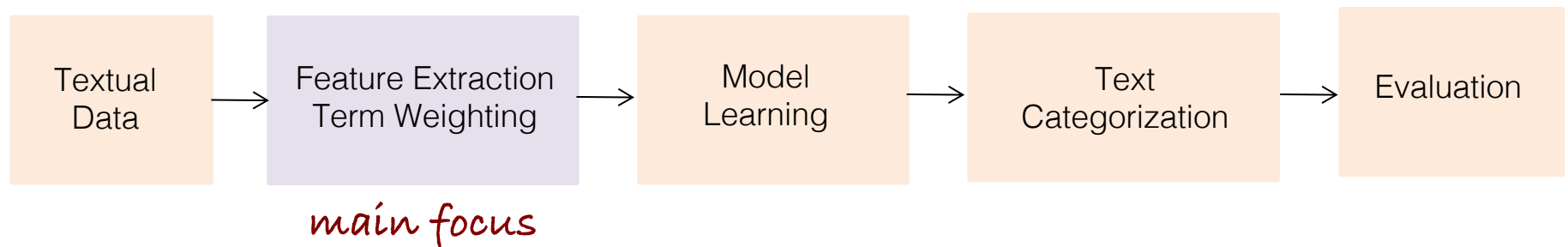
Summary

- Graphs have been widely used as modeling tools in
 - NLP
 - Text Mining
 - Information Retrieval
- Graph based event detection
 - Presentation of recent methods based 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

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



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

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?

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)

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 \Rightarrow 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 positive 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)

<div>Dataset</div> <div>Method</div>	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).

Effectiveness Results (2/2)

<div>Dataset</div> <div>Method</div>	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).

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

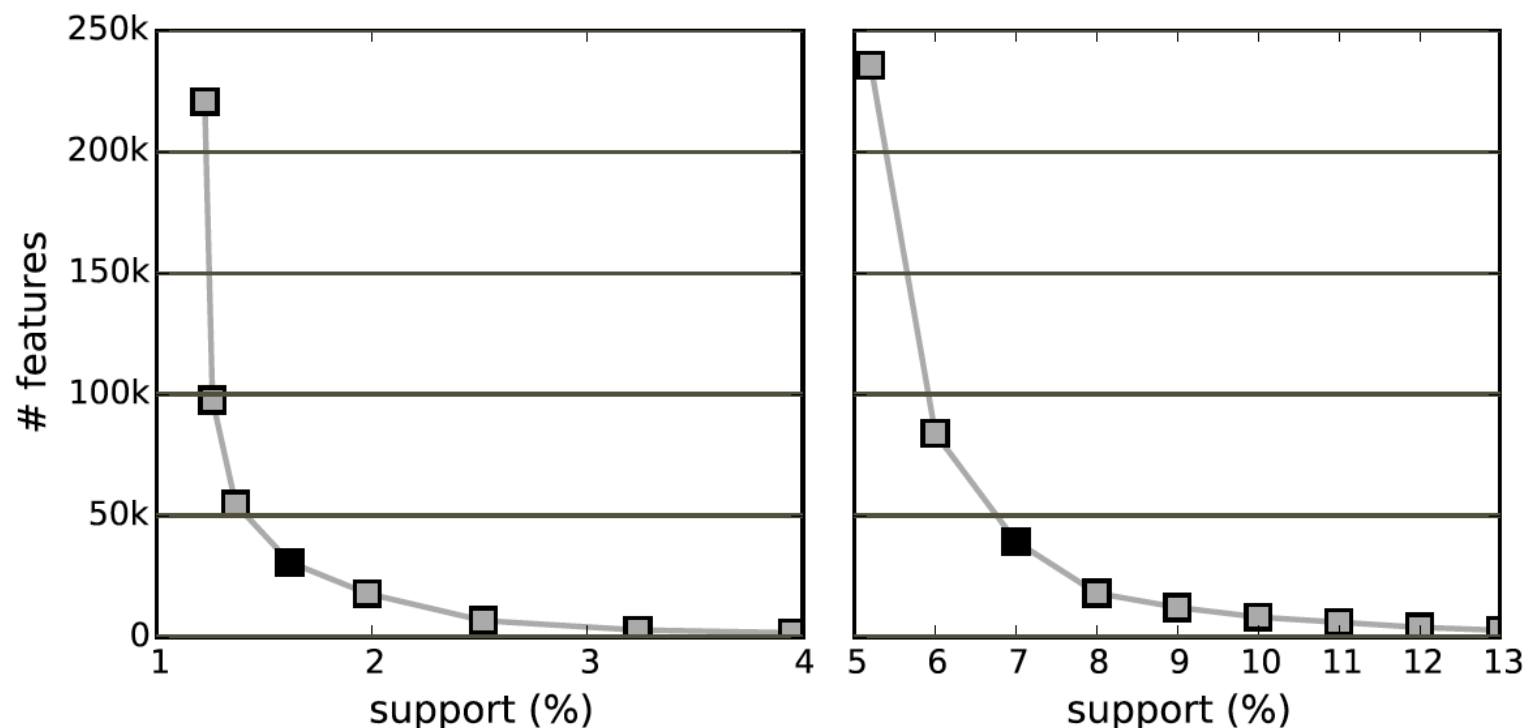


Figure: Number of subgraph features per support (%) on WebKB (left) and R8 (right) datasets. In black, the selected support chosen via the elbow method.

Distribution of Mined n-grams

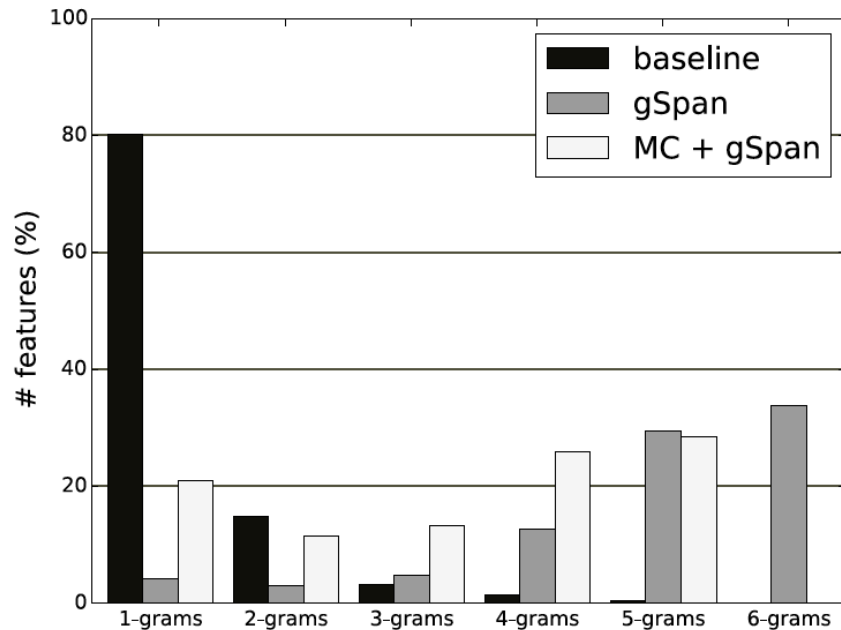


Figure: Distribution of n-grams (standard and long-distance ones) among all the features on WebKB dataset

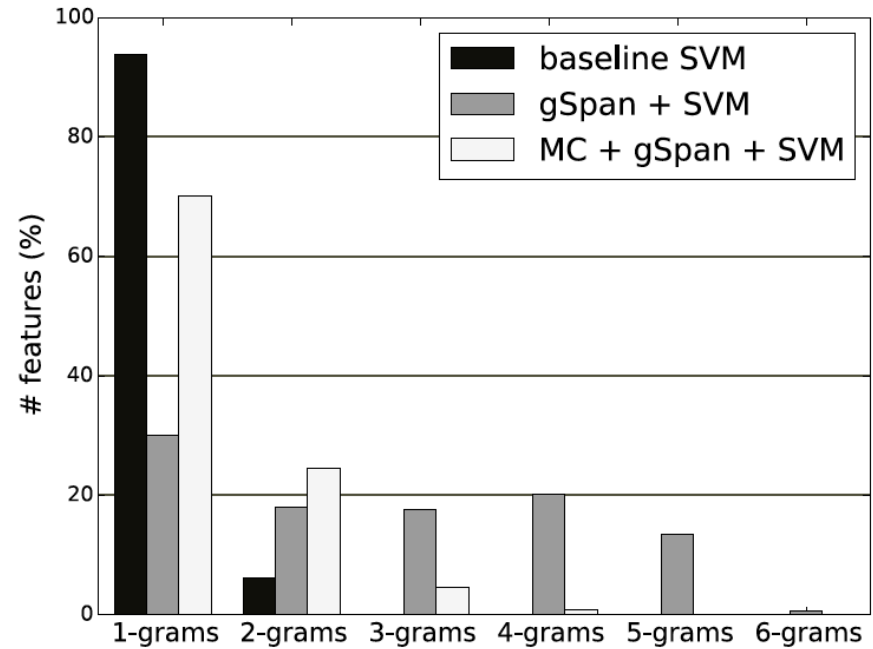


Figure: Distribution of n-grams (standard and long-distance ones) among the top 5% most discriminative features for SVM on WebKB dataset

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

Shortest-Path Graph Kernels for Document Similarity

Bag Of Words

Traditionally, documents are represented as bag of words (BOW) vectors

- I entries correspond to terms
- I non-zero for terms appearing in the document

Example

- corpus vocabulary: {the, quick, brown, cat, fox, jumped, went, over, lazy, lion, dog}
- BOW representation of sentence: “the quick brown fox jumped over the lazy dog”

1	1	1	0	1	1	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---

- However, BOW representation disregards word order!!!

Bag of n-grams

n-gram: a contiguous sequence of n words

For example, the previous sentence: “the quick brown fox jumped over the lazy dog” contains the following tri-grams:

- | | |
|---------------------|--------------------|
| 1. the quick brown | 5. jumped over the |
| 2. quick brown fox | 6. over the lazy |
| 3. brown fox jumped | 7. the lazy dog |
| 4. fox jumped over | |

n-grams distinguish word order, however, they are *too strict*

- I unlikely that the same sequence of n word appears in independent documents

Graph of Words [Rousseau, 2013]

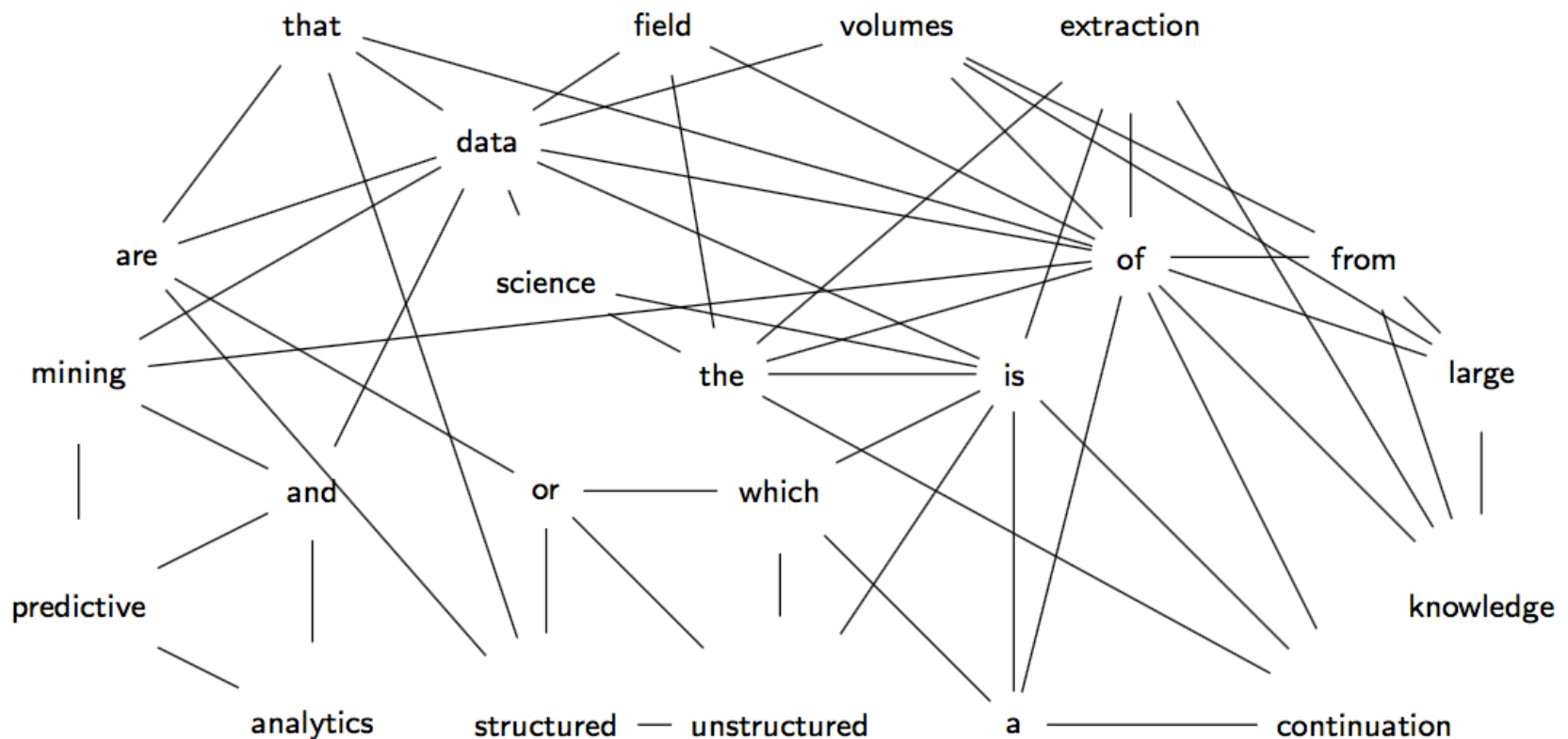
- Each document is represented as a graph $G = (V, E)$ consisting of a set V of vertices and a set E of edges between them
- vertices \rightarrow unique terms (i.e. pre-processed words)
- edges \rightarrow co-occurrences within a fixed-size sliding window
no edge weight
- no edge direction

Graph representation more flexible than n-grams. It takes into account

- word inversion
- subset matching

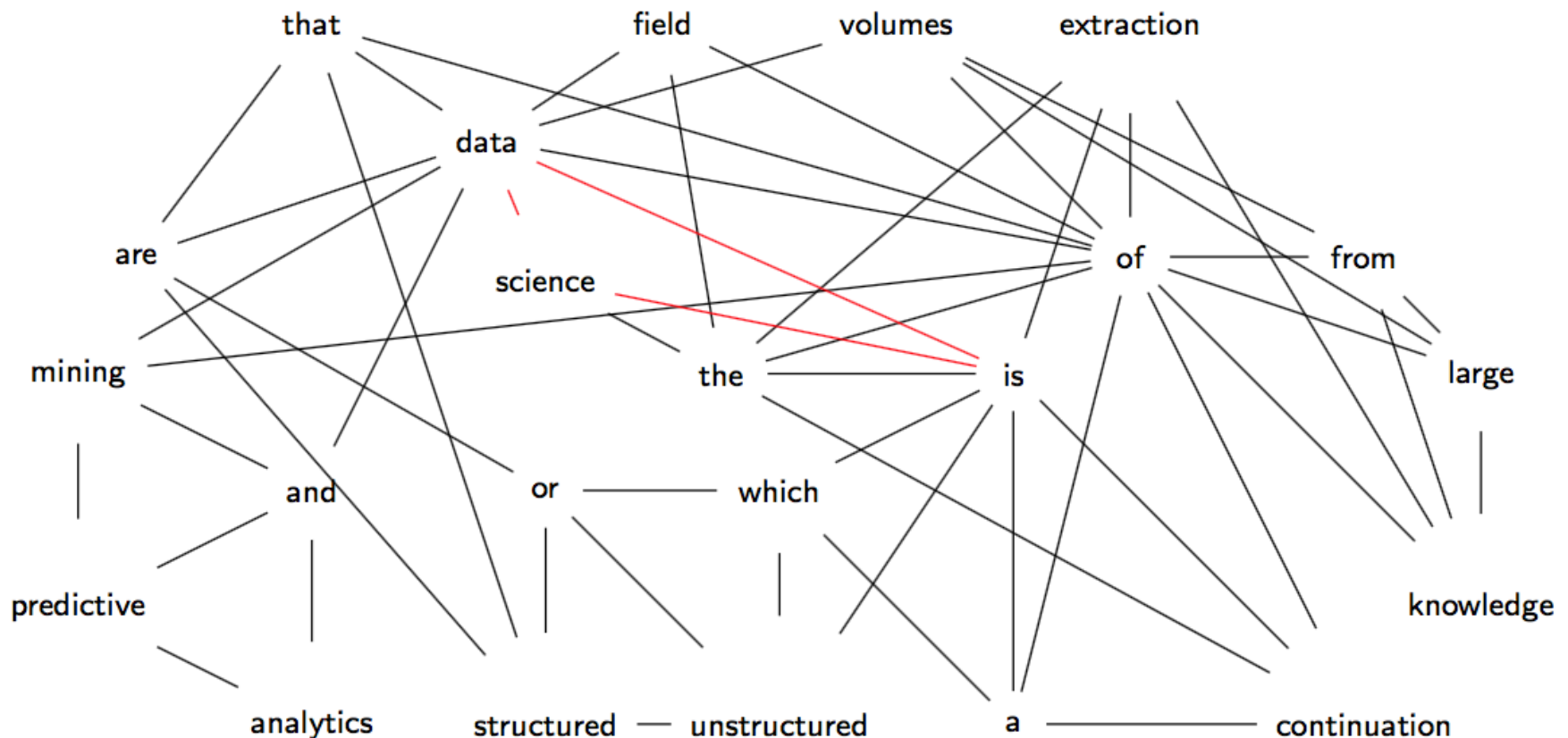
Graph of Words Example

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



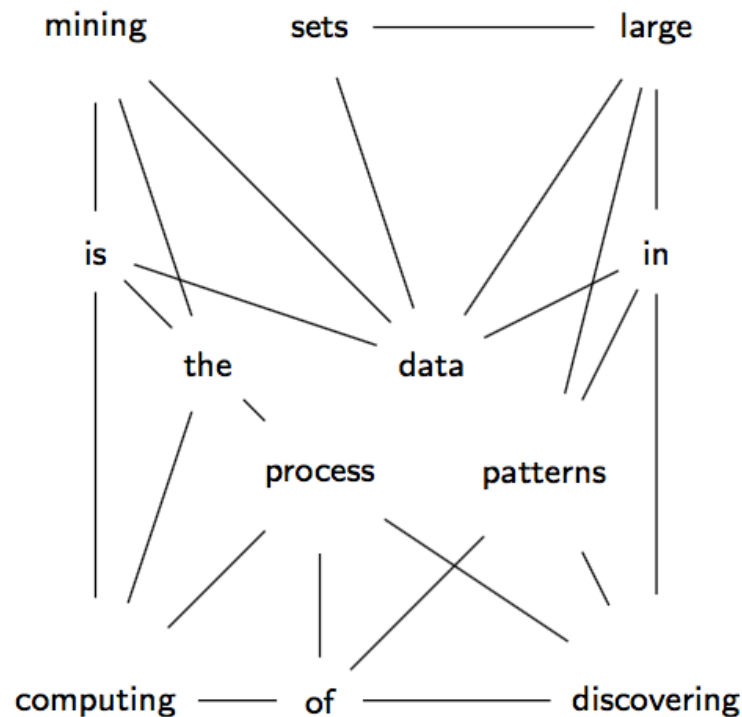
Graph of Words Example

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

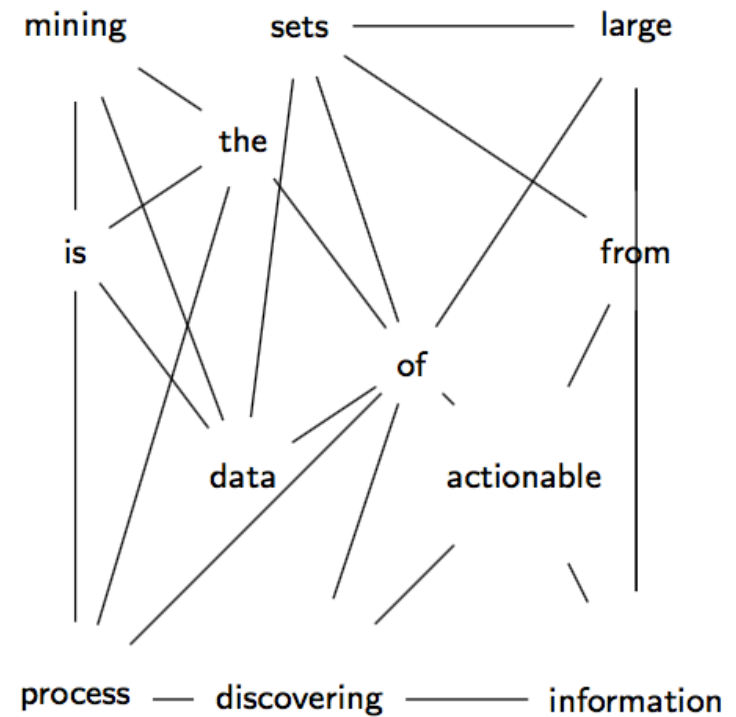


Document Similarity as Graph Comparison

Data mining is the computing process of discovering patterns in large data sets



Data mining is the process of discovering actionable information from large sets of data



Hence, **document** similarity problem → **graph** comparison problem

Applications of Graph Comparison

- Function prediction of chemical compounds
- Structural comparison and function prediction of protein structures
- Comparison of social networks
- Comparison of UML diagrams
- Document similarity

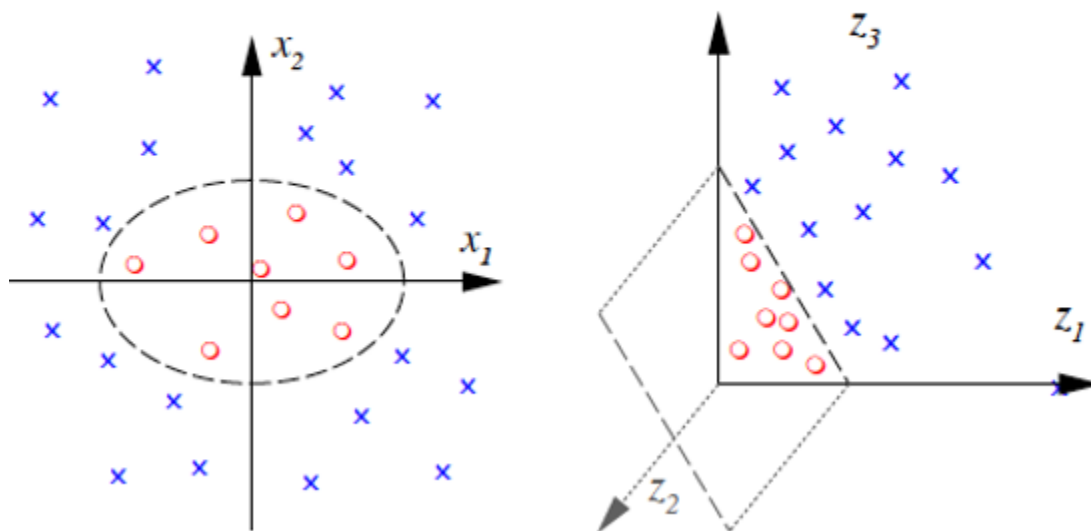
Approaches for Comparing Graphs

1. Graph isomorphism: find a mapping of the vertices of G_1 to the vertices of G_2 s.t. G_1 and G_2 are identical
 - No polynomial-time algorithm is known
 - Neither is it known to be NP-complete
2. Subgraph isomorphism: find if any subgraph of G_1 is isomorphic to a smaller graph G_2
 - NP-complete
3. Graph edit distance: count necessary operations to transform G_1 into G_2
 - Contains subgraph isomorphism check as one intermediate step
4. Graph kernels: compare substructures of graphs
 - Computable in polynomial time

Kernel Trick

The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary

$$\Phi : R^2 \rightarrow R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



Kernels for Graphs

Graph representation of objects is advantageous compared to vectorial approaches :

- able to model relations between different parts of an object as well as the values of object properties
- The dimensionality of graphs is not fixed

Problem: How to compute kernels for structured data:

- Sequences
- Graphs

Custom Shortest Path Kernel

Based on the Shortest Path Kernel proposed by [Borgwardt and Kriegel, 2005]

Compares the length of shortest paths having the same source and sink labels in two graphs-of-words

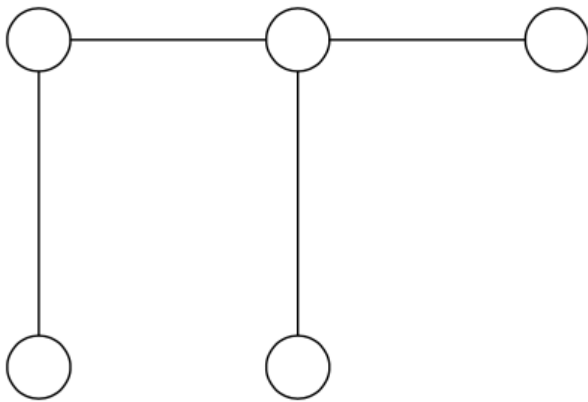
SP-transformation

Transforms the original graphs into shortest-paths graphs

- Compute the shortest-paths that are not greater than a variable d between all pairs of vertices of the input graph G using some algorithm (i. e. depth-first search)
- Create a shortest-path graph C which contains the same set of nodes as the input graph G
- All nodes which are connected by a path of length at most d in G are linked with an edge in C
- The label of an edge in C is set equal to $1/p$ where p is the length of the shortest path between its endpoints in G

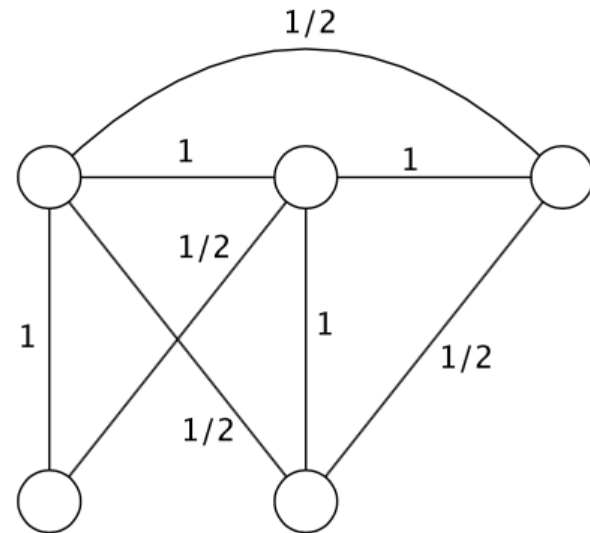
Example

SP-transformation ($d = 2$)



G

→



C

Custom Shortest Path Kernel

Given the Floyd-transformed graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ of G_1 and G_2 , the shortest path kernel is defined as:

$$k(G_1, G_2) = \frac{\sum_{v_1 \in V_1, v_2 \in V_2} k_{node}(v_1, v_2) + \sum_{e_1 \in E_1, e_2 \in E_2} k_{walk}^{(1)}(e_1, e_2)}{norm}$$

where k_{node} is a kernel for comparing two vertices, $k_{walk}^{(1)}$ a kernel on edge walks of length 1 and $norm$ a normalization factor. Specifically:

$$k_{node}(v_1, v_2) = \begin{cases} 1 & \text{if } \ell(v_1) = \ell(v_2), \\ 0 & \text{otherwise} \end{cases}$$

$$k_{walk}^{(1)}(e_1, e_2) = k_{node}(u_1, u_2) \times k_{edge}(e_1, e_2) \times k_{node}(v_1, v_2)$$

$$k_{edge}(e_1, e_2) = \begin{cases} \ell(e_1) \times \ell(e_2) & \text{if } e_1 \in E_1 \wedge e_2 \in E_2, \\ 0 & \text{otherwise} \end{cases}$$

Custom Shortest Path Kernel

Given the Floyd-transformed graphs $C_1 = (V_1, E_1)$ and $C_2 = (V_2, E_2)$ of G_1 and G_2 , the shortest path kernel is defined as:

$$k(G_1, G_2) = \frac{\sum_{v_1 \in V_1, v_2 \in V_2} k_{node}(v_1, v_2) + \sum_{e_1 \in E_1, e_2 \in E_2} k_{walk}^{(1)}(e_1, e_2)}{norm}$$

where k_{node} is a kernel for comparing two vertices, $k_{walk}^{(1)}$ a kernel on edge walks of length 1 and $norm$ a normalization factor. Specifically:

$$\mathbf{M}_1 = \mathbf{A}_1 + \mathbf{D}_1$$

$$\mathbf{M}_2 = \mathbf{A}_2 + \mathbf{D}_2$$

$$norm = \|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F$$

where $\mathbf{A}_1, \mathbf{A}_2$ are the adjacency matrices of the Floyd-transformed graphs, $\mathbf{D}_1, \mathbf{D}_2$ diagonal matrices with diagonal entries set to 1 if the corresponding term exists in the corresponding document and $\|\cdot\|_F$ is the Frobenius norm for matrices

Run Time Complexity

Standard kernel computation:

- All shortest paths from root: $\mathcal{O}(b^d)$ time (average branching factor b with breadth-first search)
- All shortest paths for n nodes: $\mathcal{O}(nb^d)$ time
- Compare all pairs of shortest paths from C_1 and C_2 : $\mathcal{O}(n^4)$
- However, since each node has a unique label, we have to consider n^2 pairs of edges
- Hence, total complexity: $\mathcal{O}(n^2 + nb^d)$

Special case for $d = 1$:

$$k(d_1, d_2) = \frac{\sum \mathbf{M}_1 \circ \mathbf{M}_2}{\|\mathbf{M}_1\|_F \times \|\mathbf{M}_2\|_F}$$

$\mathcal{O}(n + m)$ time in the worst case scenario

Evaluation

Document classification

- Goal: classify documents in a predefined set of categories
- Compute kernel matrix for all pairs of documents
- Support Vector Machine Model using the kernel matrices
- Compute Accuracy and F1 score

Link Detection

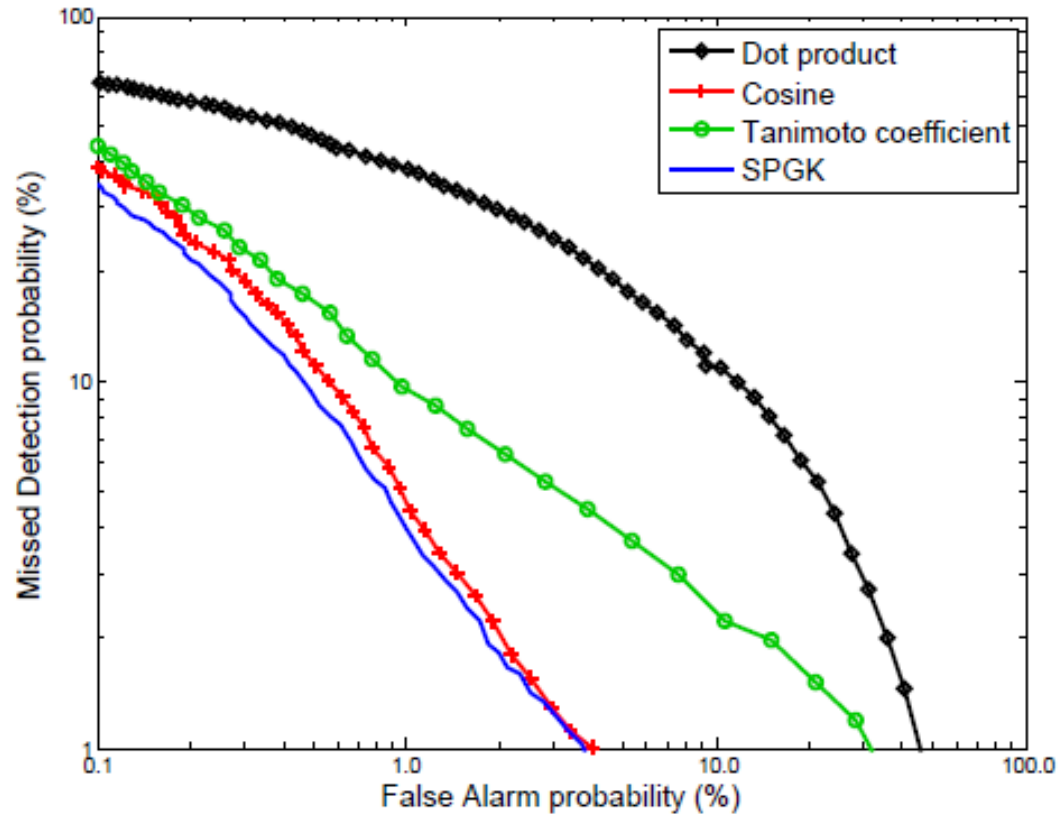
- Goal: find if two documents are linked
- Compute kernel distances for all pairs of documents
- Calculate Detection and Miss probabilities for every possible detection threshold (DET curve)

Classification results

Dataset Method		WebKB		News		Subjectivity		Amazon		Polarity	
		Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score	Accuracy	F1-score
Dot product	$n = 1$	0.9026	0.8923	0.8110	0.7764	0.8992	0.8992	0.9188	0.9188	0.7627	0.7626
	$n = 2$	0.9047	0.8950	0.8091	0.7732	0.9101	0.9101	0.9200	0.9202	0.7746	0.7745
	$n = 3$	0.9026	0.8917	0.8072	0.7710	0.9090	0.9090	0.9181	0.9185	0.7741	0.7740
	$n = 4$	0.8940	0.8813	0.8031	0.7651	0.9039	0.9039	0.9131	0.9133	0.7719	0.7718
Cosine	$n = 1$	0.9248	0.9188	0.8117	0.7766	0.9003	0.9002	0.9400	0.9400	0.7670	0.7669
	$n = 2$	0.9305	0.9275	0.8149	0.7797	0.9094	0.9094	0.9413	0.9413	0.7756	0.7756
	$n = 3$	0.9298	0.9259	0.8097	0.7738	0.9099	0.9099	0.9419	0.9418	0.7765	0.7765
	$n = 4$	0.9248	0.9208	0.8076	0.7709	0.9076	0.9075	0.9413	0.9413	0.7753	0.7753
Tanimoto	$n = 1$	0.9062	0.8983	0.8155	0.7815	0.9094	0.9093	0.9225	0.9226	0.7749	0.7748
	$n = 2$	0.9040	0.8945	0.8075	0.7700	0.9061	0.9060	0.9181	0.9185	0.7735	0.7735
	$n = 3$	0.9241	0.9180	0.7980	0.7575	0.9021	0.9020	0.9344	0.9347	0.7648	0.7648
	$n = 4$	0.9176	0.9084	0.7899	0.7483	0.8953	0.8952	0.9300	0.9300	0.7586	0.7586
DCNN		0.8918	0.8799	0.7991	0.7615	0.9026	0.9026	0.9181	0.9181	0.7326	0.7326
CNN	static,rand	> 1 day		0.7757	0.7337	0.8716	0.8715	0.8881	0.8882	0.7150	0.7150
	non-static,rand	> 1 day		0.8113	0.7749	0.8961	0.8960	0.9356	0.9356	0.7654	0.7653
SPGK	$d = 1$	0.9327	0.9278	0.8104	0.7749	0.9148	0.9148	0.9400	0.9401	0.7776	0.7775
	$d = 2$	0.9370	0.9336	0.8089	0.7729	0.9146	0.9146	0.9413	0.9413	0.7789	0.7788
	$d = 3$	0.9291	0.9233	0.8078	0.7703	0.9137	0.9137	0.9444	0.9444	0.7761	0.7760
	$d = 4$	0.9291	0.9223	0.8097	0.7730	0.9118	0.9118	0.9463	0.9463	0.7780	0.7780

- On 4/5 datasets, SPGK outperforms the other three similarity measures and the NN architectures

Story Link Detection

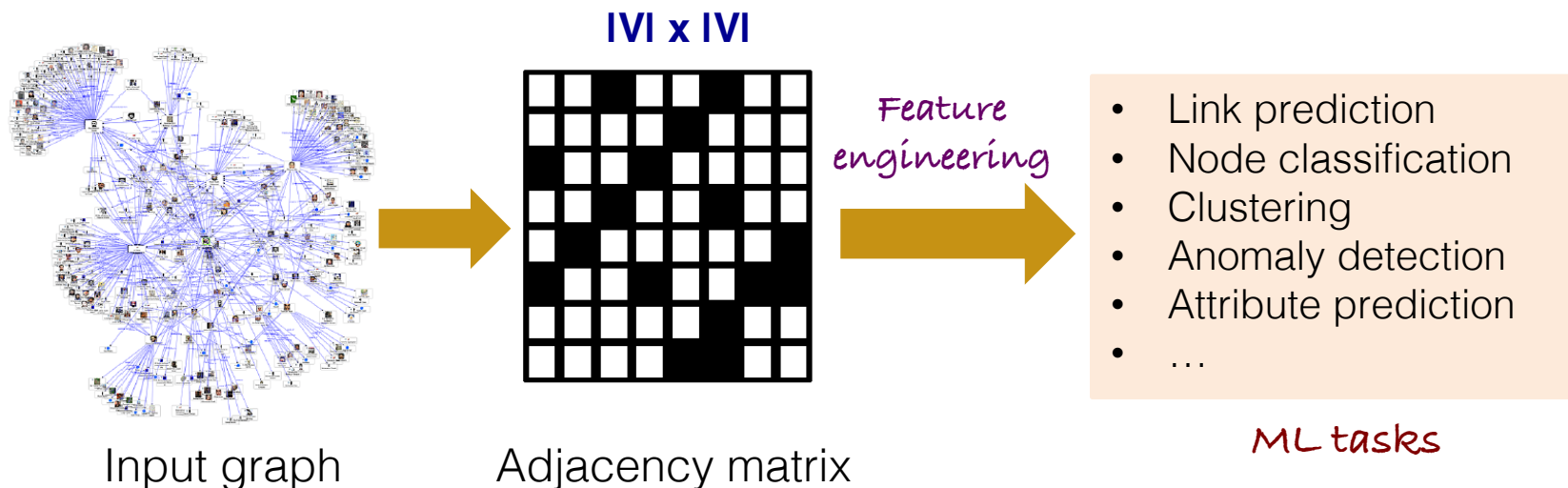


Graph representation learning with applications in NLP

(text categorization and word analogy)

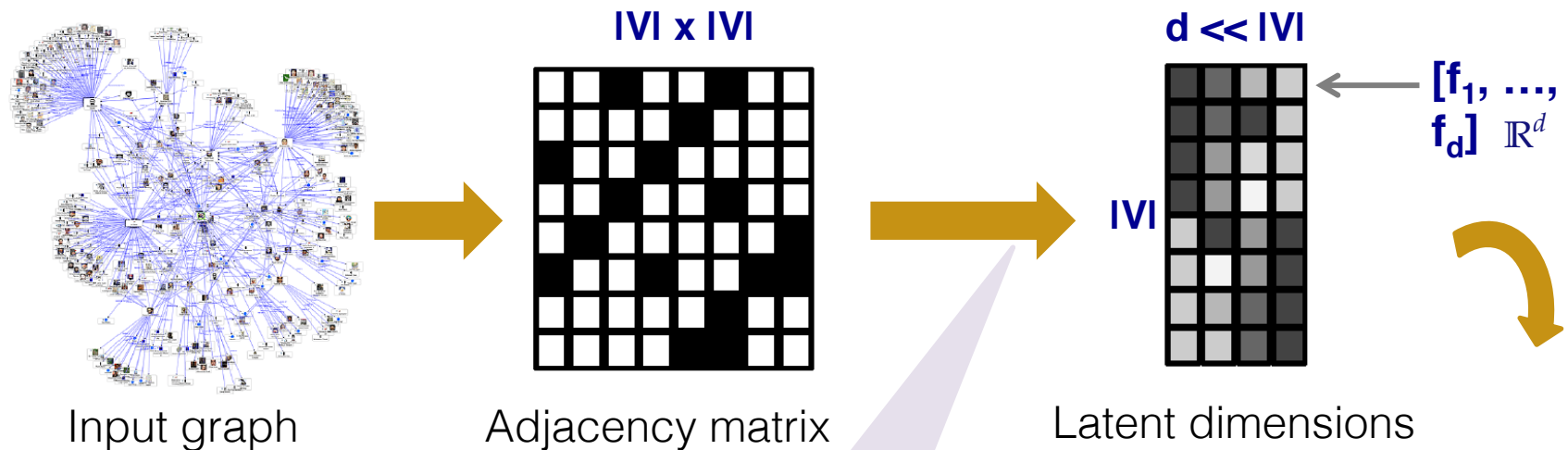
Feature Extraction From Graphs

- 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

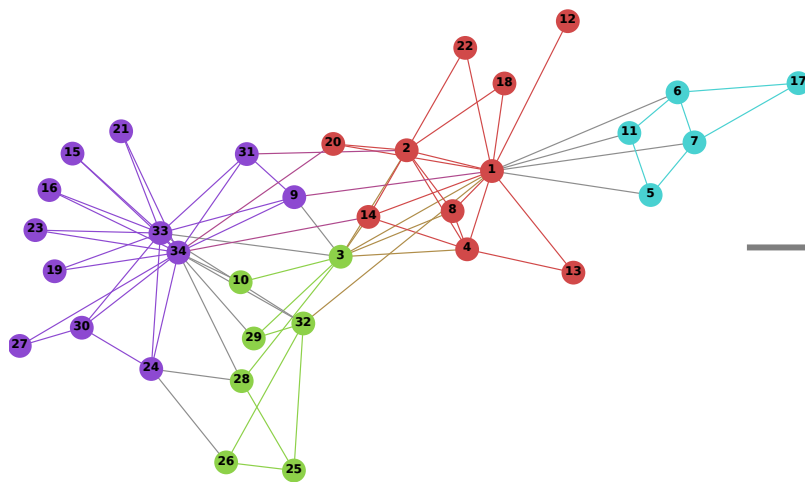
- Create features by transforming the graph into a lower dimensional latent representation



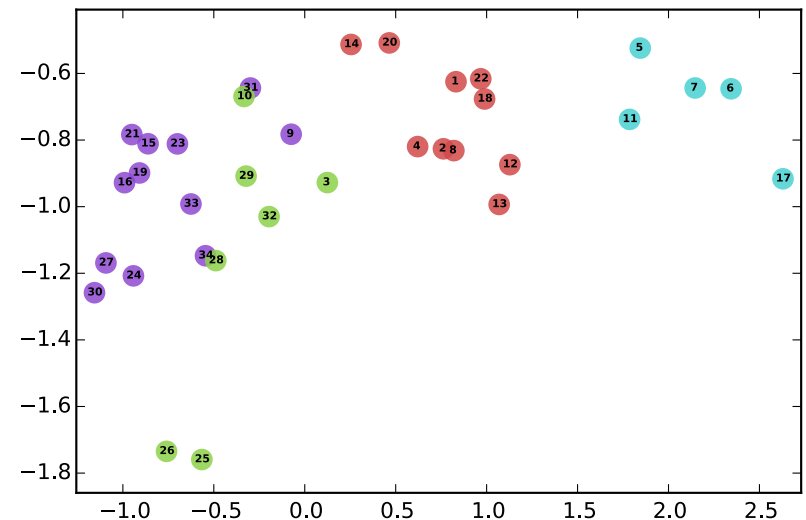
How to learn a latent representation of a graph?

- Link prediction
- Node classification
- Clustering
- Anomaly detection
- Attribute prediction
- ... *ML tasks*

Example: Community Detection



Input graph



Learn latent representation

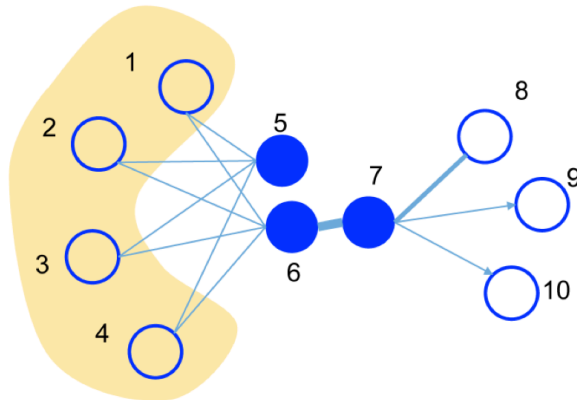
Other applications: classification, link prediction, ...

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

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

LINE with First-order Proximity (1/2)

Model the probability of an edge (i, j) between \mathbf{v}_i and \mathbf{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 \mathbf{u}_i

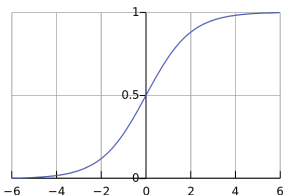
Joint probability between \mathbf{u}_i and \mathbf{u}_j

Original (graph) space

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

Empirical distribution over
the space $\mathbf{V} \times \mathbf{V}$

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 \mathbf{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 \mathbf{v}_i , we model the conditional distribution $\mathbf{p}_2(\bullet|\mathbf{v}_i)$ over all the “contexts” (all the nodes in the network)
- **Assumption of second-order proximity:** Nodes with similar distributions $\mathbf{p}_2(\bullet|\mathbf{v}_i)$ over the “contexts” are similar

LINE with Second-order Proximity (2/3)

For directed edge (\mathbf{i}, \mathbf{j}) , model the probability of context \mathbf{v}_j generated by node \mathbf{v}_i (i.e., probability of an edge from \mathbf{v}_i to \mathbf{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 \mathbf{v}_i

Conditional distribution $\mathbf{p}_2(\bullet|\mathbf{v}_i)$ over the contexts

Original (graph) space

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

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

Empirical distribution over the space $\mathbf{V} \times \mathbf{V}$

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

Experiments – Word Analogy

- Language network: word analogy

“Paris”

 - Find solution to (“China”, “Beijing” \rightarrow “France, “?”)
 - Given word embeddings, find word \mathbf{d}^* whose embedding \mathbf{u}_d is closest to vector $\mathbf{u}_{\text{Beijing}} - \mathbf{u}_{\text{China}} + \mathbf{u}_{\text{France}}$

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**

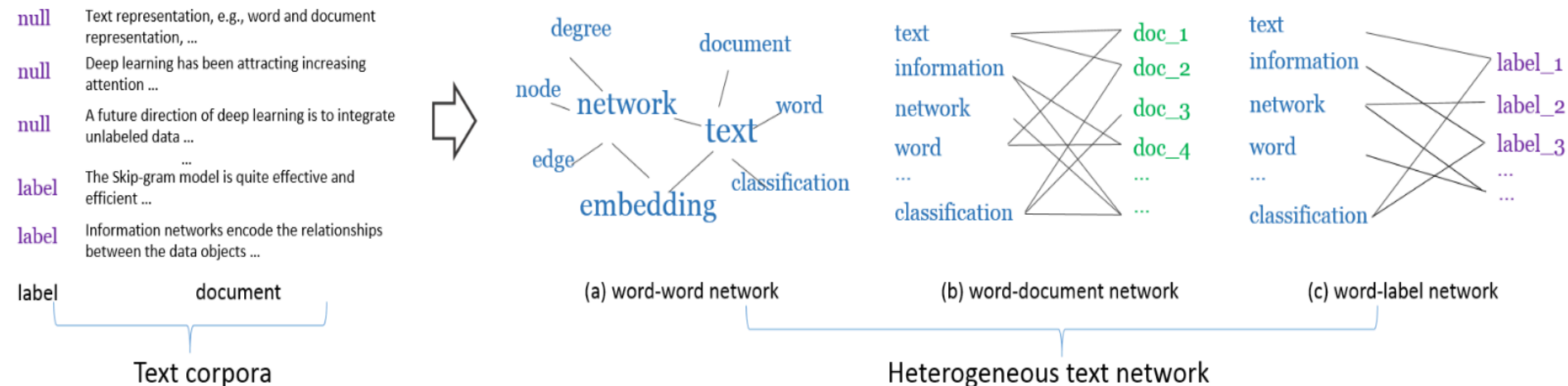
- Classification of Wikipedia articles
 - Choose articles from 7 categories
- How to obtain the document vectors for classification?
 - Average of the corresponding word vector representations

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks

- Unsupervised text embedding methods(Skip-gram, Paragraph Vector): simplicity, scalability, and effectiveness.
- performance inferior to deep learning architectures (i.e. CNNs) for machine learning tasks.
- possible reasons:
 - text embedding learn representation of text in a fully unsupervised way,
 - No leveraging the labeled information available for the task.
 - low dimensional representations learned not particularly tuned for any task.
- Predictive text embedding (PTE): semi-supervised representation learning method
- utilizes both labeled and unlabeled data to learn the text embedding.
- Labeled information and different levels of word co-occurrence information represented as a *large-scale heterogeneous text network*, embedded into a low dimensional space via an efficient algorithm.
- The low dimensional embedding preserves
 - semantic closeness of words and documents,
 - strong predictive power for the particular task.
- Compared to recent supervised approaches based on CNNs, predictive text embedding is
 - more effective, much more efficient, fewer parameters to tune

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks

- utilize both labeled and unlabeled data to learn the text embedding.
- Labeled information and different levels of word co-occurrence information represented as a *large-scale heterogeneous text network*



PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks -

Bipartite Network Embedding

- Assume a bipartite graph $V_a \rightarrow V_b, i/j$ from V_a/V_b resp.
- u_i, u_j embeddings of vertices i, j . Prob they are connected:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}$$

- conditional distribution $p(\cdot|v_j)$ over all the vertices in the set V_a ; for each pair of vertices $v_j, v_{j'}$,
- second-order proximity can actually be determined by their conditional distributions: $p(\cdot|v_j) p(\cdot|v_{j'})$

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks -

Bipartite Network Embedding

- Preserve second-order proximity:
 - conditional distribution $p(\cdot|v_j)$ be close to empirical distribution $\hat{p}(\cdot|v_j)$

$$O = \sum_{j \in B} \lambda_j d(\hat{p}(\cdot|v_j), p(\cdot|v_j)).$$

- $d(\cdot, \cdot)$: KL-divergence between two distributions, λ_j : importance of vertex v_j in the network (degree)
- Omitting constants:

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

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks -

Bipartite Network Embedding

- Preserve second-order proximity:

- conditional distribution $p(\cdot|v_j)$ empirical distribution \hat{p}

$$O = \sum_{j \in B} \lambda_j d(\hat{p}(\cdot|v_j), p(\cdot|v_j)).$$

- $d(\cdot, \cdot)$: KL-divergence between two distributions, λ_j : importance of vertex v_j in the network (degree)

- Omitting constants:
$$O = - \sum_{(i,j) \in E} w_{ij} \log p(v_j|v_i).$$

- optimized with stochastic gradient descent with
 - edge sampling: a binary edge $e = (i,j)$ is sampled with the probability proportional to its weight w_{ij} ,
 - Negative sampling: multiple negative edges (i,j) are sampled from a noise distribution.

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks -

Heterogeneous Network Embedding

- For each network conditional distribution $p(u_i|u_j)$, $p(u_i|d_j)$, $p(u_i|l_j)$
- The global optimization: $O_{pte} = O_{ww} + O_{wd} + O_{wl}$,

- Where:

$$O_{ww} = - \sum_{(i,j) \in E_{ww}} w_{ij} \log p(v_i|v_j)$$

$$O_{wd} = - \sum_{(i,j) \in E_{wd}} w_{ij} \log p(v_i|d_j)$$

$$O_{wl} = - \sum_{(i,j) \in E_{wl}} w_{ij} \log p(v_i|l_j)$$

- Training: joint vs. Pre-training + Fine-tuning

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks - Experiments

Table 1: Statistics of the Data Sets

	Long Documents							Short Documents		
Name	20NG	WIKI	IMDB	CORPORATE	ECONOMICS	GOVERNMENT	MARKET	DBLP	MR	TWITTER
Train	11,314	1,911,617*	25,000	245,650	77,242	138,990	132,040	61,479	7,108	800,000
Test	7,532	21,000	25,000	122,827	38,623	69,496	66,020	20,000	3,554	400,000
V	89,039	913,881	71,381	141,740	65,254	139,960	64,049	22,270	17,376	405,994
Doc. length	305.77	672.56	231.65	102.23	145.10	169.07	119.83	9.51	22.02	14.36
#classes	20	7	2	18	10	23	4	6	2	2

*In the WIKI data set, only 42,000 documents are labeled.

Table 2: Results of text classification on **long** documents.

		20NG		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	80.88	79.30	79.95	80.03	86.54	86.54
Unsupervised Embedding	Skip-gram	70.62	68.99	75.80	75.77	85.34	85.34
	PVDBOW	75.13	73.48	76.68	76.75	86.76	86.76
	PVDM	61.03	56.46	72.96	72.76	82.33	82.33
	LINE(G_{ww})	72.78	70.95	77.72	77.72	86.16	86.16
	LINE(G_{wd})	79.73	78.40	80.14	80.13	89.14	89.14
	LINE($G_{ww} + G_{wd}$)	78.74	77.39	79.91	79.94	89.07	89.07
Predictive Embedding	CNN	78.85	78.29	79.72	79.77	86.15	86.15
	CNN(pretrain)	80.15	79.43	79.25	79.32	89.00	89.00
	PTE(G_{wl})	82.70	81.97	79.00	79.02	85.98	85.98
	PTE($G_{ww} + G_{wl}$)	83.90	83.11	81.65	81.62	89.14	89.14
	PTE($G_{wd} + G_{wl}$)	84.39	83.64	82.29	82.27	89.76	89.76
	PTE(pretrain)	82.86	82.12	79.18	79.21	86.28	86.28
	PTE(joint)	84.20	83.39	82.51	82.49	89.80	89.80

PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks - Experiments

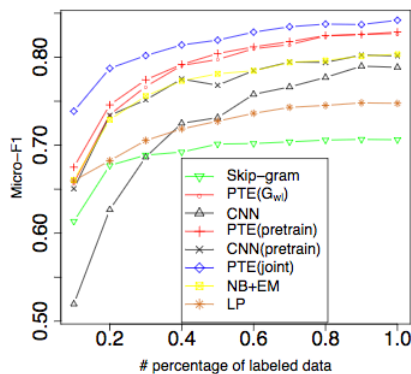
Table 3: Results of text classification on **long** documents (RCV1 data sets).

Algorithm	Corporate		Economics		Government		Market	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
BOW	78.45	63.80	86.18	81.67	77.43	62.38	95.55	94.09
PVDBOW	65.87	45.78	79.63	74.82	70.74	54.08	91.81	88.88
LINE(G_{wd})	76.76	60.30	85.55	81.46	77.82	63.34	95.66	93.90
PTE(G_{wl})	76.69	60.48	84.88	80.02	78.26	63.69	95.58	93.84
PTE(pretrain)	77.03	61.03	84.95	80.63	78.48	64.50	95.54	93.79
PTE(joint)	79.20	64.29	87.05	83.01	79.63	66.15	96.19	94.58

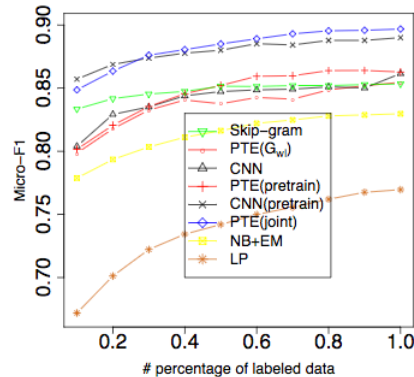
Table 4: Results of text classification on **short** documents.

Type	Algorithm	DBLP		MR		Twitter	
		Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Word	BOW	75.28	71.59	71.90	71.90	75.27	75.27
Unsupervised Embedding	Skip-gram	73.08	68.92	67.05	67.05	73.02	73.00
	PVDBOW	67.19	62.46	67.78	67.78	71.29	71.18
	PVDM	37.11	34.38	58.22	58.17	70.75	70.73
	LINE(G_{ww})	73.98	69.92	71.07	71.06	73.19	73.18
	LINE(G_{wd})	71.50	67.23	69.25	69.24	73.19	73.19
	LINE($G_{ww} + G_{wd}$)	74.22	70.12	71.13	71.12	73.84	73.84
Predictive Embedding	CNN	76.16	73.08	72.71	72.69	75.97	75.96
	CNN(pretrain)	75.39	72.28	68.96	68.87	75.92	75.92
	PTE(G_{wl})	76.45	72.74	73.44	73.42	73.92	73.91
	PTE($G_{ww} + G_{wl}$)	76.80	73.28	72.93	72.92	74.93	74.92
	PTE($G_{wd} + G_{wl}$)	77.46	74.03	73.13	73.11	75.61	75.61
	PTE(pretrain)	76.53	72.94	73.27	73.24	73.79	73.79
	PTE(joint)	77.15	73.61	73.58	73.57	75.21	75.21

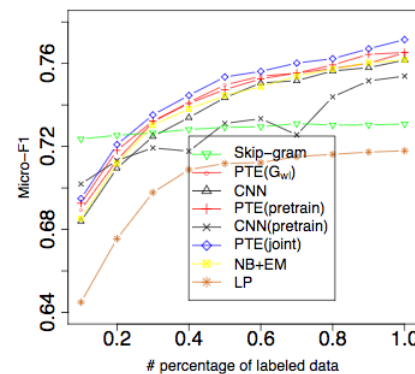
PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks - Experiments



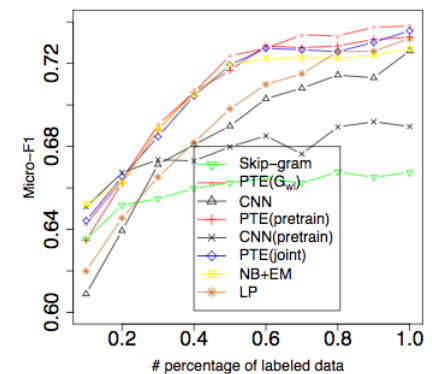
(a) 20NG



(b) IMDB

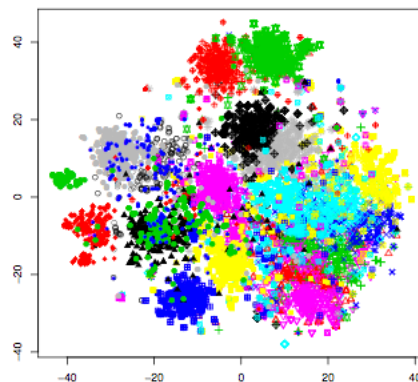


(c) DBLP

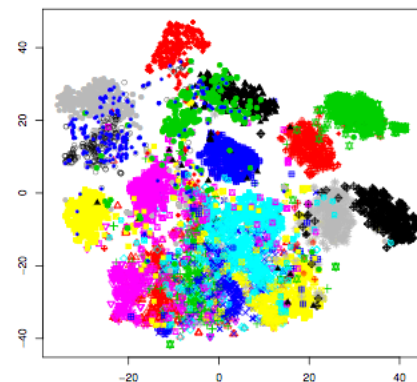


(d) MR

Performance with regards to labeled data



(c) Test(LINE(G_{wd}))



(d) Test(PTE(G_{wl}))

document visualization using unsupervised and predictive embeddings on 20ng data set

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**

CentraleSupélec and Inria

fragkiskos.malliaros@centralesupelec.fr

<http://fragkiskos.me>



- **Michalis Vazirgiannis**

Data Science and Mining group

École Polytechnique, France

mvazirg@lix.polytechnique.fr

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

<https://www.lix.polytechnique.fr/dascim/>



Material: www.lix.polytechnique.fr/~mvazirg/gow_tutorial_webconf_2018.pdf

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, Francois 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).

References – Event detection

- **Tweeting rate:** Volume of status updates
 - Nichols, J., Mahmud, J., Drews, C.: Summarizing Sporting Events Using Twitter. In: IUI. pp. 189–198 (2012)
 - Zhao, S., Zhong, L., Wickramasuriya, J., Vasudevan, V.: Human as Real-Time Sensors of Social and Physical Events: A Case Study of Twitter and Sports Games. arXiv:1106.4300 (2011)
 - Zubiaga, A., Spina, D., Amigó, E., Gonzalo, J.: Towards Real-time Summarization of Scheduled Events from Twitter Streams. In: HT. pp. 319–320 (2012)
- **Retweet rate:** Volume of retweets updates
 - Chierichetti, F., Kleinberg, J., Kumar, R., Mahdian, M., Pandey, S.: Event Detection via Communication Pattern Analysis. In: ICWSM. pp. 51–60 (2014)

References – Event detection

- **Modified Hidden Markov Model:** tweet rate and the word distribution
 - Chakrabarti, D., Punera, K.: Event Summarization Using Tweets. In: ICWSM. Pp. 66–73 (2011)
- **Mixture model:** detect sub-events for each participant
 - Shen, C., Liu, F., Weng, F., Li, T.: A Participant-based Approach for Event Summarization Using Twitter Streams. In: NAACL-HLT. pp. 1152–1162 (2013)
- **Hierarchical Dirichlet processes:** probabilistic topic model
 - Srijith, P., Hepple, M., Bontcheva, K., Preotiuc-Pietro, D.: Sub-story detection in twitter with hierarchical dirichlet processes. Information Processing & Management (2016)
- **Graph degeneracy:** sequences of tweets as graphs
 - Meladianos, P., Nikolentzos, G., Rousseau, F., Stavarakas, Y., Vazirgiannis, M.: Degeneracy-based Real-Time Sub-Event Detection in Twitter Stream. In: ICWSM.pp. 248–257 (2015)