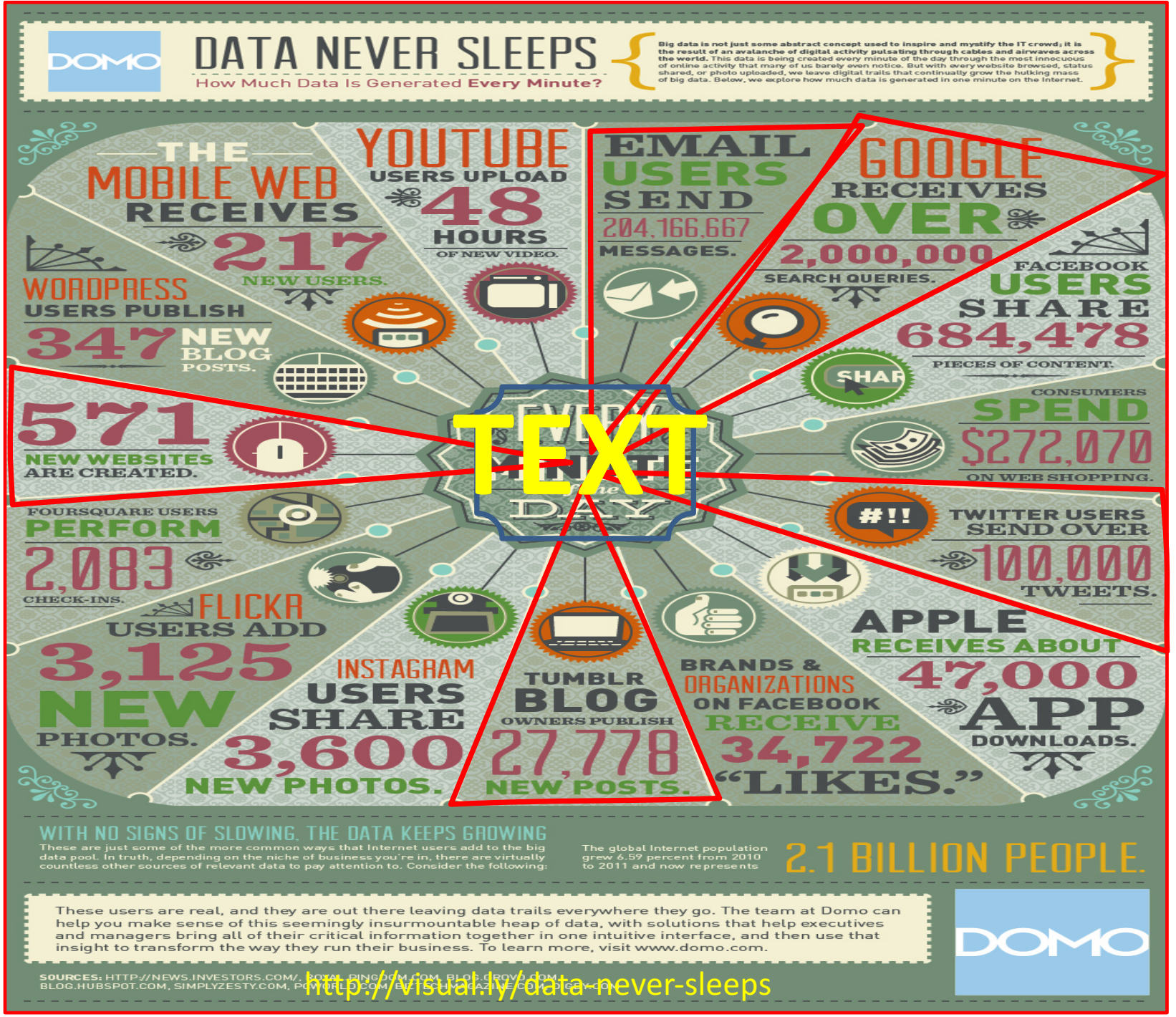
Text Mining – an introduction

Michalis Vazirgiannis

LIX @ Ecole Polytechnique

January 2017


DOMO
DATA NEVER SLEEPS
How Much Data Is Generated Every Minute?
Big data is not just some abstract concept used to inspire and mystify the IT crowd; it is the result of an avalanche of digital activity pulsating through cables and airwaves across the world. This data is being created every minute of the day through the most innocuous of online activity that many of us barely even notice. But with every website browsed, status shared, or photo uploaded, we leave digital trails that continually grow the hulking mass of big data. Below, we explore how much data is generated in one minute on the Internet.
THE MOBILE WEB RECEIVES 217 NEW USERS.
YOUTUBE USERS UPLOAD 48 HOURS OF NEW VIDEO.
EMAIL USERS SEND 204,166,667 MESSAGES.
GOOGLE RECEIVES OVER 2,000,000 SEARCH QUERIES.
FACEBOOK USERS SHARE 684,478 PIECES OF CONTENT.
WORDPRESS USERS PUBLISH 347 NEW BLOG POSTS.
571 NEW WEBSITES ARE CREATED.
TEXT
CONSUMERS SPEND $272,070 ON WEB SHOPPING.
FOURSQUARE USERS PERFORM 2,083 CHECK-INS.
TWITTER USERS SEND OVER 100,000 TWEETS.
FLICKR USERS ADD 3,125 NEW PHOTOS.
INSTAGRAM USERS SHARE 3,600 NEW PHOTOS.
TUMBLR BLOG OWNERS PUBLISH 27,778 NEW POSTS.
BRANDS & ORGANIZATIONS ON FACEBOOK RECEIVE 34,722 "LIKES."
APPLE RECEIVES ABOUT 47,000 APP DOWNLOADS.
WITH NO SIGNS OF SLOWING, THE DATA KEEPS GROWING
These are just some of the more common ways that Internet users add to the big data pool. In truth, depending on the niche of business you're in, there are virtually countless other sources of relevant data to pay attention to. Consider the following:
The global Internet population grew 6.59 percent from 2010 to 2011 and now represents 2.1 BILLION PEOPLE.
These users are real, and they are out there leaving data trails everywhere they go. The team at Domo can help you make sense of this seemingly insurmountable heap of data, with solutions that help executives and managers bring all of their critical information together in one intuitive interface, and then use that insight to transform the way they run their business. To learn more, visit www.domo.com.
DOMO
SOURCES: HTTP://NEWS.INVESTORS.COM/, BOXAL.BINGDOM.COM, BLOG.GROVO.COM, BLOG.HUBSPOT.COM, SIMPLYZESTY.COM, POWERED.COM, EVERYSECONDCOUNTS.EU
http://visual.ly/data-never-sleeps

Outline

- Document collection preprocessing
- Feature Selection
- Indexing
- Query processing & Ranking
- Retrieval evaluation

Text representation for Information Retrieval

- We seek a transformation of the textual content of documents into a vector space representation.
  - Assume documents as the data
  - Dimensions are the distinct terms used
- Example : “This is the database lab of the IS master course”

This	is	the	database	lab	of	IS	master	course

Boolean Vector Model

- Boolean model
  - Text 1: “This is the database lab of the IS master course”
  - Text 2: “This is a database course”

	This	is	a	the	database	lab	of	IS	master	course
Text 1:	1	1	0	1	1	1	1	1	1	1
Text 2:	1	1	1	0	1	0	0	0	0	1

Vector Space Model

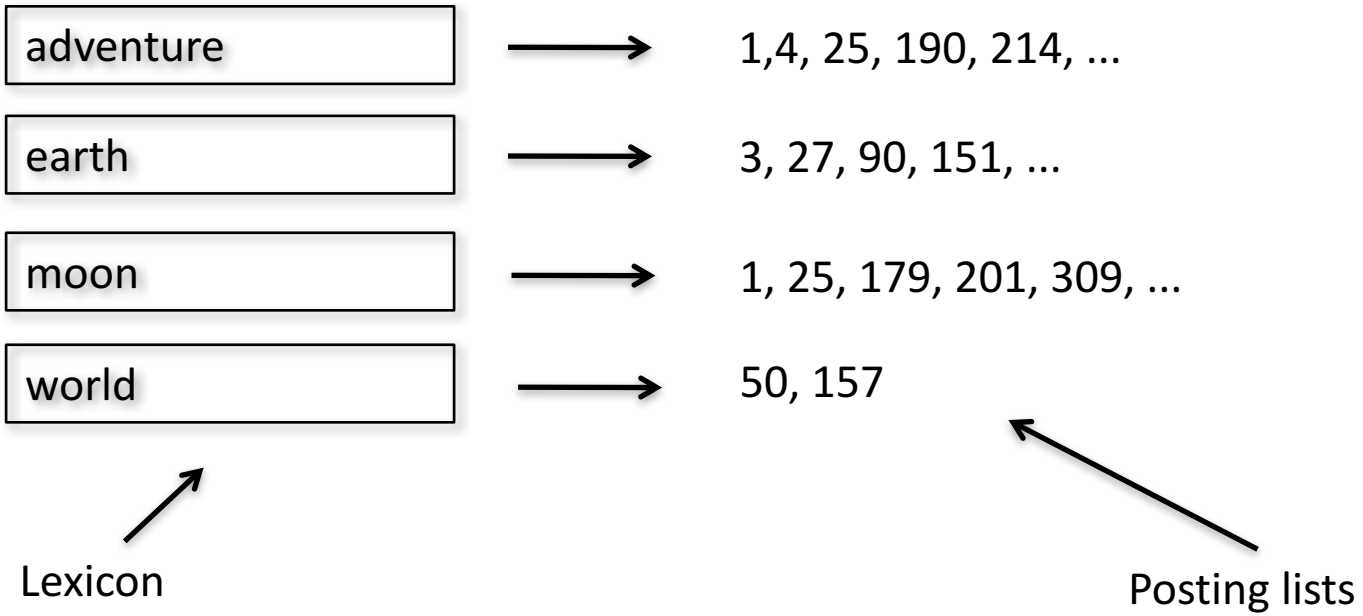
- Vector Space Model:
  - To VSM represents the significance of each term for each document
    - The cell values are computed based on the terms' frequency
    - Most common approach is TF/IDF

Feature Selection

	This	is	a	the	database	lab	of	IS	master	course
Text 1:	0.1	0.1	0	0.2	0.1	0.1	0.1	0.1	0.1	0.1
Text 2:	0.2	0.2	0.2	0	0.2	0	0	0	0	0.2

Inverted Index

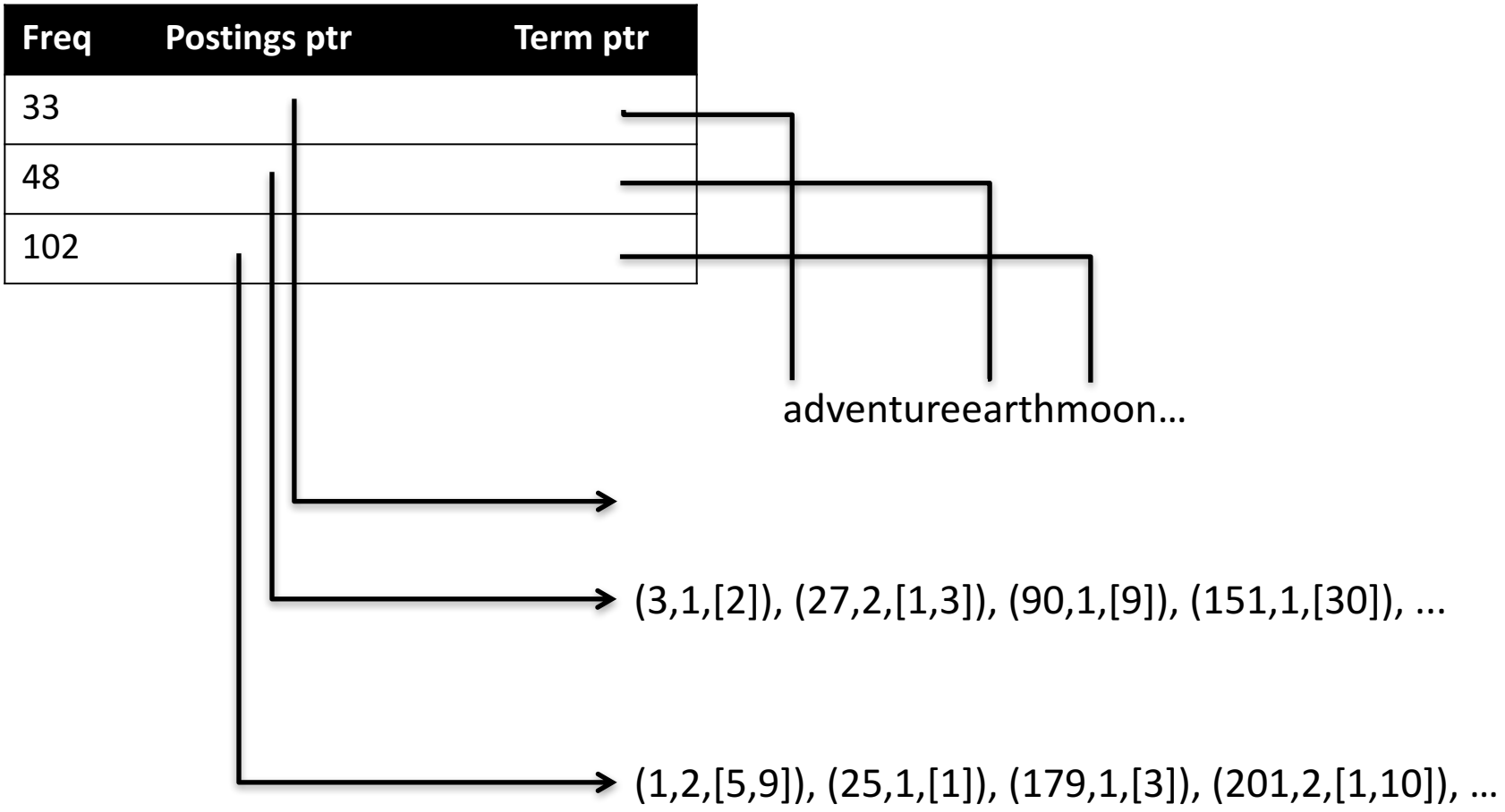
- Record the documents in which each term occurs in
  - Similar to the *Index* at the end of books



Lexicon structures

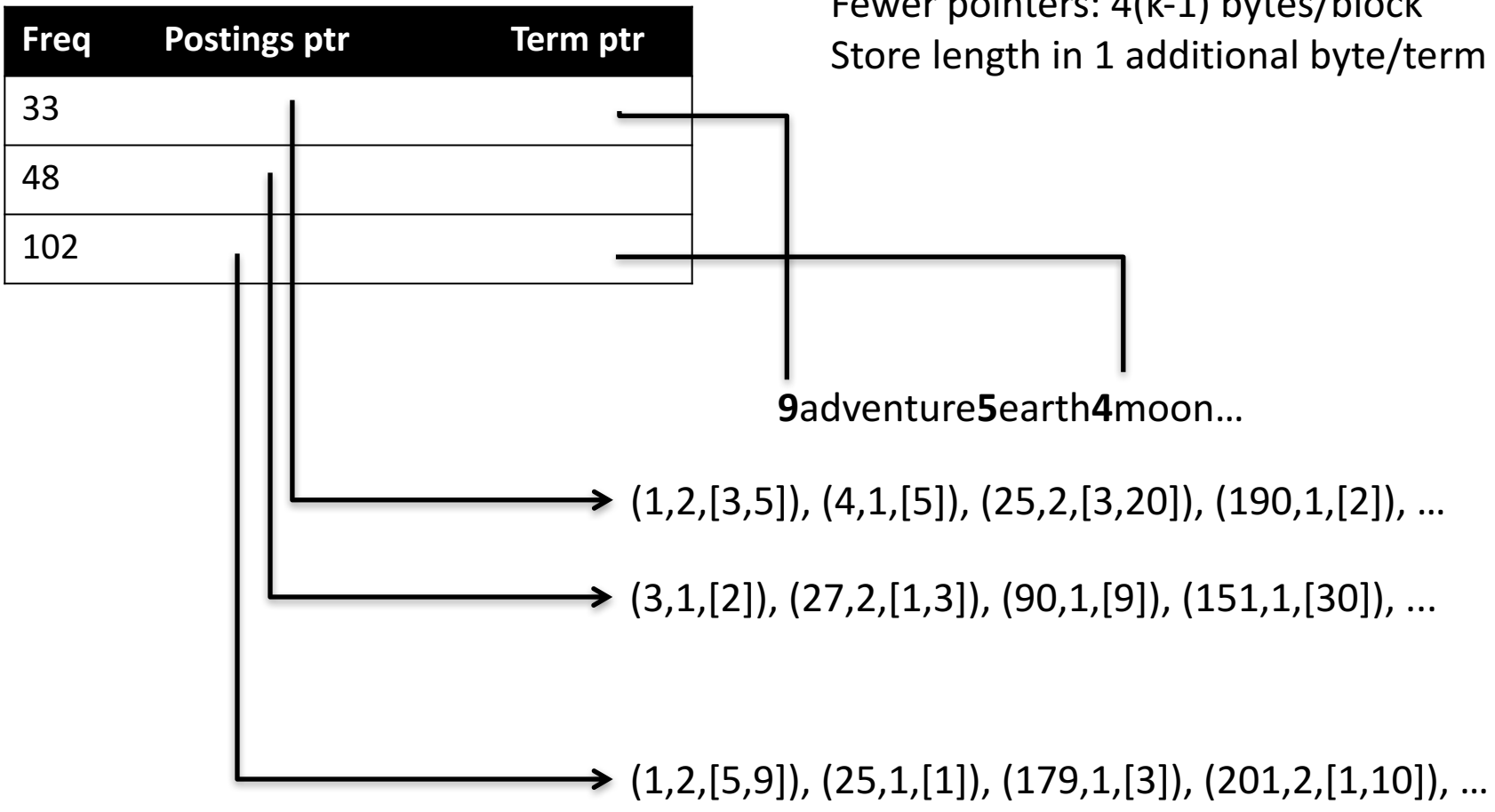
- Fixed-size entry array, allocating
  - *n* bytes for storing the term string
  - 8 bytes for pointing to the posting list
  - 32 bytes for storing statistics and numerical term ids
- Needs total *N* x (*n*+40) bytes per entry
  - If *n*=20, *N*=50M we need 2.79Gb for the lexicon
- Allocating more space than needed
  - Average length of English words is approx. 4.5

Dictionary as a String



Dictionary as a String with Blocks

Store pointers every k terms
Fewer pointers: 4(k-1) bytes/block
Store length in 1 additional byte/term



Front-coding

- Further compress strings within the same block
- Sorted terms share common prefix
  - store only the differences

8automata8automate9automatic10automation

→8automat*a1:e2:ic3:ion

Payload in posting lists

- Store linguistic information in posting lists

```
A  recent event at the National Library in Athens, drew a  crowd of 300.
```

Payload in posting lists

- Store linguistic information in posting lists
  - Part-of-Speech information per term occurrence

```
A  recent event at the National Library in Athens, drew a  crowd of 300.
DT JJ     NN    IN DT  NNP      NNP     IN NNP     VDB  DT NN    IN CD
```

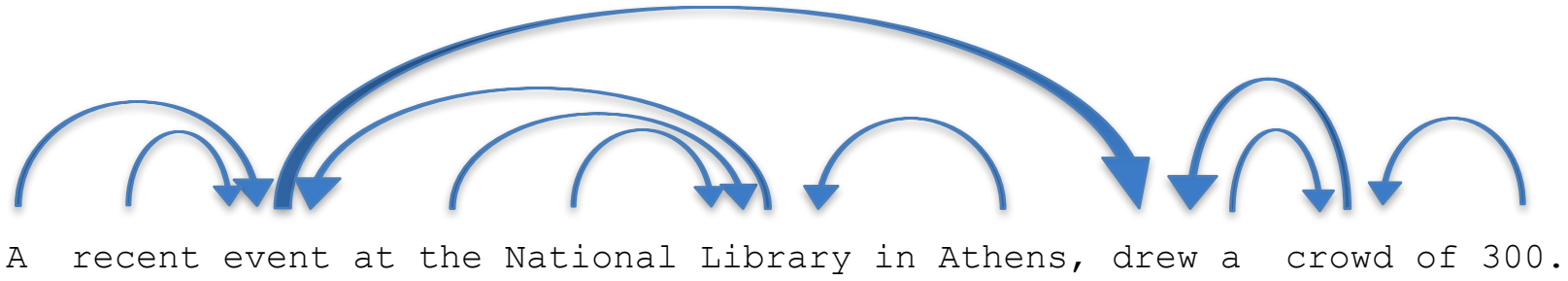
Payload in posting lists

- Store linguistic information in posting lists
  - Part-of-Speech information per term occurrence
  - References to named entities

A	recent	event	at	the	National	Library	in	Athens,	drew	a	crowd	of	300.
O	O	O	O	O	**ORG**	**ORG**	O	**LOC**	O	O	O	O	O

Payload in posting lists

- Store linguistic information in posting lists
  - Part-of-Speech information per term occurrence
  - References to named entities
  - Dependency parse trees



Inverted Index construction

- Steps:
  - Document processing:
    - Parsing, Tokenization, Linguistic processing
  - Inversion of posting lists

Document parsing

- Handle different document formats
  - Html, PDF, MS Word, Flash, PowerPoint, …
- Detect encoding of characters
  - How to translate bytes to characters?
  - Popular choices for Web pages:
    - UTF-8, ISO8859-7, Windows 1253
- Detect language of text
  - Estimate the probability of sequences of characters from a sample of documents
  - Assign the most likely language to an unseen document

Tokenization

- Split text in sequences of tokens which are candidates to be indexed
- But, pay attention to
  - Abbreviations: **U.N.** and **UN** (United Nations or 1 in French)
  - **New York** as one or two tokens
  - **c++** as one token, but not c+
  - Apostrophes
  - Hyphenation: one-man-show, Hewlett-Packard
  - Dates: 2011/05/16, May 16th, 2011
  - Numbers: (+33) 8203-911
  - Accents: Ελλάδα / Ελλαδα, Université

Stop-words

- Very frequent words that do not carry semantic information
  - the, a, an, and, or, to
- Stop-words can be removed to reduce index size requirements and to speed-up query processing

- But
  - Improvements in compression and query processing can offset the impact from stop-words
  - Stop-words are useful for certain queries submitted to Web search engines
    - “The The”, “Let it be”, “To be or not to be”

Lemmatization & Stemming

- Lemmatization
  - Reduce inflected forms of a word so that they are treated as a single term: am, were, being, been → be
  - Requires knowledge of context, grammar, part of speech
- Stemming: reduces tokens to a "root" form
  - Porter's Stemming Algorithm
    - Applicable to texts written in English
    - Removes the longest-matching suffix from words

      EED → EE agreed →agree, **but** feed →feed

      ED → plastered→plaster, **but** bled → bled

      ING → motoring→motor, **but**sing→sing

    - Limitation: resulting terms are not always readable

Sort-Based Indexing

- Steps
  - Collect all pairs term-docID from documents
  - Sort pairs on term and then docID
  - Organize the docIDs for each term in posting lists
- Optimization
  - Map each term to termID and work with pairs termID-docID
- Limitation
  - Not enough memory to hold termID-docID pairs
  - External sort algorithm

Single-Pass In-Memory Index Construction

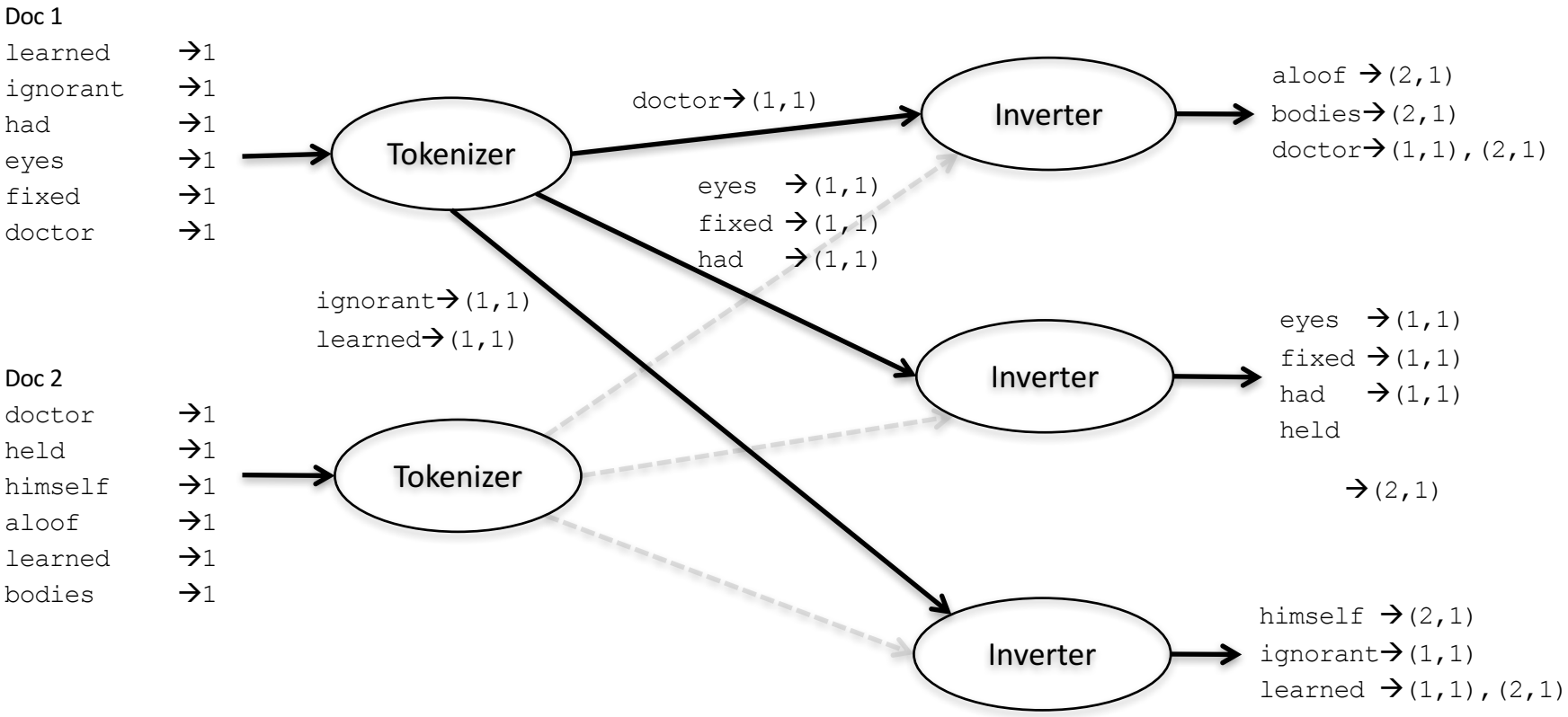
- Main idea
  - Build intermediate complete inverted indexes
  - At the end, merge the intermediate indexes
  - No need to keep information between intermediate indexes

```
While more docs to process
    Initialize dictionary
    While free memory available
        Get next token
        If term(token) exists in dictionary
            Then get posting_list
            Else add new posting_list to dictionary
        Add term(token), docID to posting_list
    Sort dictionary terms
    Write block of postings to disk
Merge blocks of postings
```

Distributed Indexing

- Single-Pass In-Memory indexing can be applied for any number of documents
  - **But** it will take too long to index 100 billion Web pages
- Indexing can be parallelized
  - Clusters of commodity servers used to index billions of documents

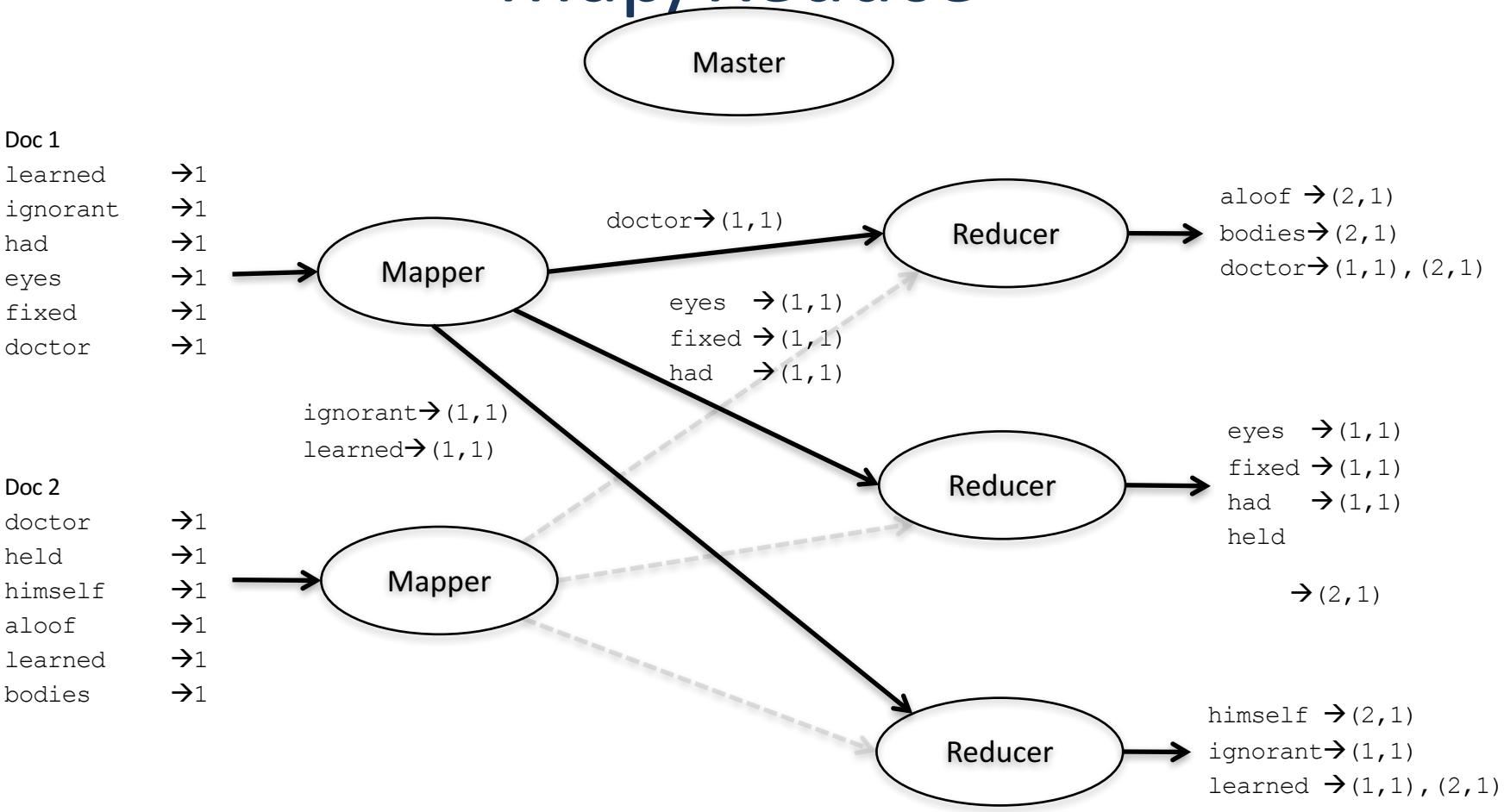
Distributed Indexing



Map-Reduce framework

- Framework for handling distribution transparently
  - provides distribution, replication, fault-tolerance
- Computation is modeled as a sequence of **Map** and **Reduce** steps
  - **Map** emit (key, value) pairs
  - **Reduce** collects (key, value) pairs for a range of keys

Distributed Indexing with Map/Reduce



Mapper emits pairs: term -> (docid, frequency)
Reducer emits pairs: term -> posting list

Queries & Document ranking

Query-document matching scores

- How do we compute the score of a query-document pair?
- one-term query: *"Sentiment"*
- If the term *"Sentiment"* does not occur in the document: score=0.
- The more frequent the query term in the document, the higher the score
- We will look at a number of alternatives for doing this.

28

Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let *A* and *B* be two sets
- Jaccard coefficient: $Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$

$(A \neq \emptyset \text{ or } B \neq \emptyset)$

- Jaccard (*A*, *A*) = 1 - Jaccard (*A*, *B*) = 0 if $A \cap B = 0$
- A and B don't have to be the same size.

Example

What is the query-document match score that the Jaccard coefficient computes for:

- Query: "ides of March"
- Document "Caesar died in March"
- JACCARD(*q*, *d*) = 1/6

Issues with Jaccard?

- It doesn't consider term frequency
- Rare terms are more informative than frequent terms. Jaccard does capture this.
- We need a more sophisticated way of normalizing for the length of a document.

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Problems with Jaccard:

- doesn't consider term frequency - Rare terms are more informative than frequent terms.

31

Frequency incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

32

Bag of words model

- We do not consider the order of words in a document.
- *"Mike is heavier than Molly"* and *"Molly is heavier than Mike"* are represented the same way.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.

33

Term Frequency (1)

- **term frequency** tf(*t*,*d*),
  - simplest choice: use the *raw frequency* of a term in a document, i.e. tf(*t*,*d*) = f(*t*,*d*): the number of times that term *t* occurs in document *d*:
  - Other possibilities:
- boolean "frequencies": tf(*t*,*d*) = 1 if *t* occurs in *d* and 0 otherwise;
- logarithmically scaled frequency: tf(*t*,*d*) = 1 + log f(*t*,*d*) (and 0 when f(*t*,*d*) = 0);
- normalized frequency,

$$tf(t,d) = \frac{f(t,d)}{\max\{f(w,d) : wind\}}$$

- raw frequency divided by the maximum raw frequency of any term in the document.
  - prevent a bias towards longer documents,.

Term frequency: Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$: 0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, ...
- Score for a query document (q,d) pair:

  tf-matching-score(q, d) = $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$

- The score is 0 if none of the query terms is present in the document.

Frequency in document vs. frequency in collection

In addition, to term frequency we care about term frequency at collection level for weighting and ranking.

Rare terms are more informative than frequent terms.

- Consider a rare query term (e.g., Hypermnesia).
- A document containing this term is very likely to be relevant.
- We want high weights for rare terms like Hypermnesia

Frequent terms

- are less informative (i.e. GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- For frequent terms like GOOD, INCREASE and LINE, we assign positive weights but lower than for rare ones.

Document frequency

- high weights for rare terms like Hypermnesia
- low (positive) weights for frequent words like GOOD, INCREASE and LINE.
- Factor document frequency into the matching score.
- The document frequency df_t is # of documents in the collection that the term occurs in.
- df_t is an inverse measure of the informativeness of term *t*.
- We define the idf weight of term t as: $\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$

  (*N*: # documents in the collection.)
- idf_t is a measure of the informativeness of the term.
- [log N/df_t] instead of [N/df_t] to "dampen" the effect of idf

Examples for idf

- Compute idf_t using the formula:

$$\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of idf on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query "*hypermensia feature*", idf weighting increases the relative weight of "*hypermensia*" and decreases the relative weight of "*feature*".
- idf has little effect on ranking for one-term queries.

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- increases with the number of occurrences within a document. (tf)
- increases with the rarity of the term in the collection. (idf)
- Best known weighting scheme in information retrieval

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

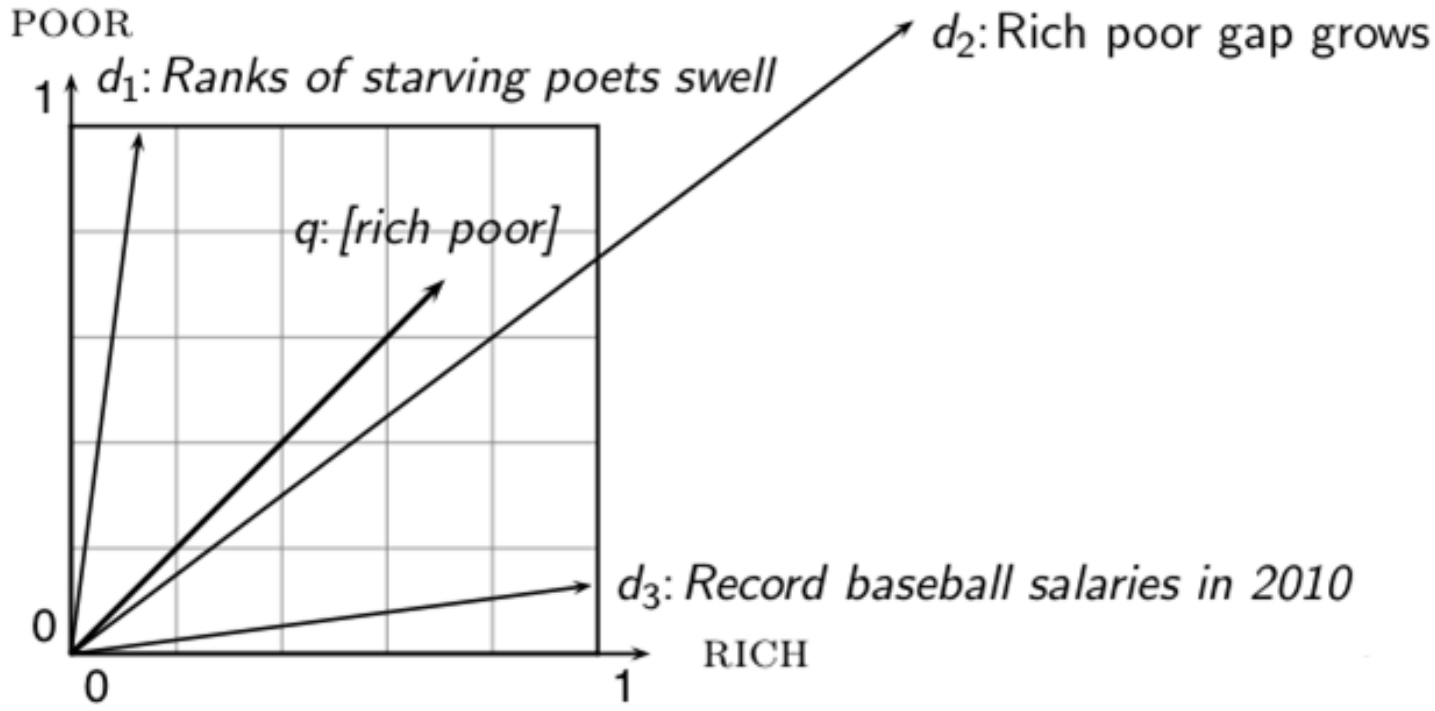
Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Binary → count → weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0
MERCY	1.51	0.0	1.90	0.12	5.25	0.88
WORSER	1.37	0.0	0.11	4.15	0.25	1.95
. . .						

Each document is now represented as a real-valued vector of tf x idf weights $\in R^{|V|}$.

Why distance is a bad idea

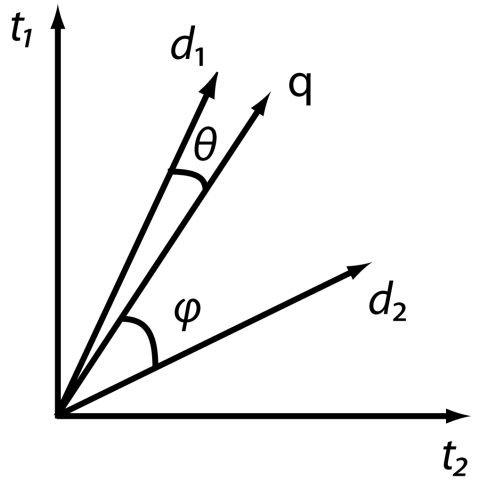


The Euclidean distance of $\vec{q}$ and $\vec{d}_2$ is large although the distribution of terms in the query *q* and the distribution of terms in the document d_2 are very similar.

Vector Space Model

- Document ***d*** and query ***q*** are represented as k-dimensional vectors $d = (w_{1,d}, \ldots, w_{k,d})$ and $q = (w_{1,q}, \ldots, w_{k,q})$
  - Each dimension corresponds to a term from the collection vocabulary
  - Independence between terms
  - $w_{i,q}$ is the weight of i-th vocabulary word in ***q***
- Is Euclidean distance appropriate to measure the similarity?
  - Vector (a, b) and (10 x a, 10 x b) contain the same words but have large Euclidean distance
- Degree of similarity between ***d*** and ***q*** is the cosine of the angle between the two vectors

$$sim(d,q) = \frac{d \cdot q}{|d||q|} = \frac{\sum_{t=1}^{k} w_{t,d} \times w_{t,q}}{\sqrt{\sum_{t=1}^{k} w_{t,d}^2} \times \sqrt{\sum_{t=1}^{k} w_{t,q}^2}}$$



Term weighting in VSM

- Term weighting with tf-idf (and variations)

$$w_{t,d} = tf_{t,d} \times idf_t$$

- tf models the importance of a term in a document

$$tf_{t,d} = f_{t,d} \qquad tf_{t,d} = \frac{f_{t,d}}{\max\left(f_{s,d}\right)}$$

  - $f_{t,d}$ is the frequency of term *t* in document *d*
- idf models the importance of a term in the document collection
  - Logarithm base not important
  - Information content of event "term *t* occurs in document *d*"

$$idf_t = -\log P\left(t \text{ occurs in } d\right) = -\log \frac{n_t}{N} = \log \frac{N}{n_t} \qquad idf_t = \log \frac{N - n_t + 0.5}{n_t + 0.5}$$

  - *N* is the total number of documents, n_t is the document frequency of term *t*

Ranked retrieval in the vector space model

- Represent the query as a weighted tf-idf vector
- Represent each document as a weighted tf-idf vector
- Compute the cosine similarity between the query vector and each document vector
- Rank documents with respect to the query
- Return the top *K* (e.g., *K* = 10) to the user

46

Example - TFIDF (1)

- Doc1: Computer Science is the scientific field that studies computers
- Doc2: Decision Support Systems support enterprises in decisions
- Doc3: Information Systems are based on Computer Science

- Dictionary/Dimensions:{computer, science, field, studies, decision, support, systems, enterprises, information, based}

TF:

computer	science	field	studies	decision	support	systems	enterprises	information	based
2/6	2/6	1/6	1/6	0	0	0	0	0	0
0	0	0	0	2/6	2/6	1/6	1/6	0	0
1/5	1/5	0	0	0	0	1/5	0	1/5	1/5

IDF:

computer	science	field	studies	decision	support	systems	enterprises	information	based
0.301	0.301	0.602	0.602	0.602	0.602	0.301	0.602	0.602	0.602

Example - TFIDF (2)

TFIDF:

computer	science	field	studies	decision	support	systems	enterprises	information	based
0.1	0.1	0.1	0.1	0	0	0	0	0	0
0	0	0	0	0.2	0.2	0.05	0.05	0	0
0.06	0.06	0	0	0	0	0.06	0	0.12	0.12

Queries

- Each query is considered as a new document and there fore it can be represented as a vector. Then the k most similar documents are retrieved
- Query = {Information Systems}

Collection:

computer	science	field	studies	decision	support	systems	enterprises	information	based
0.1	0.1	0.1	0.1	0	0	0	0	0	0
0	0	0	0	0.2	0.2	0.05	0.05	0	0
0.06	0.06	0	0	0	0	0.06	0	0.12	0.12

Query:

computer	science	field	studies	decision	support	systems	enterprises	information	based
0	0	0	0	0	0	1	0	1	0

	Distance	I.P	cos(φ)
Doc 1	1.43	0	0
Doc 2	1.40	0.05	0.40
Doc 3	1.34	0.18	0.64

BM25 Ranking Function

- Ranking function assuming bag-of-words document representation

$$score(d,q) = \sum_{t \in d \cap q} idf_t \times \frac{tf_{t,d} \cdot (k_1 + 1)}{tf_{t,d} + k_1 \cdot \left(1 - b + b \frac{len_d}{avglen}\right)}$$

  - len_d is the length of document d
  - *avglen* is the average document length in the collection

- Score depends only on query terms

- Values of parameters k_1 and b depend on collection/task
  - k_1 controls term frequency saturation
  - b controls length normalization
  - Default values: $k_1 = 1.2$ and $b = 0.75$

Text retrieval evaluation

Text retrieval evaluation

- Typical evaluation setting
  - Set of documents
  - Set of information needs, expressed as queries (typically 50 or more)
  - Relevance assessments specifying for each query the relevant and non-relevant documents

Evaluating unranked results

	Relevant	Non-relevant
Retrieved	True positives (tp)	False positives (fp)
Not retrieved	False negatives (fn)	True negatives (tn)

$$\text{Precision} = \frac{\#(\text{Relevant documents retrieved})}{\#(\text{Retrieved documents})} = \frac{tp}{(tp + fp)}$$

$$\text{Recall} = \frac{\#(\text{Relevant documents retrieved})}{\#(\text{Relevant documents})} = \frac{tp}{(tp + fn)}$$

Precision/recall tradeoff

- You can increase recall by returning more docs.
- Recall is a non-decreasing function of the number of docs retrieved.
- A system that returns all docs has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.

54

A combined measure: *F*

- *F* allows us to trade off precision against recall.

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \qquad \beta^2 = \frac{1-\alpha}{\alpha}$$

- $\alpha \in [0, 1]$ and thus $\beta^2 \in [0,\infty]$
- Most frequently used: balanced *F* with $\beta = 1$ or $\alpha = 0.5$
  - This is the harmonic mean of *P* and *R*: $\frac{1}{F} = \frac{1}{2}(\frac{1}{P} + \frac{1}{R})$

55

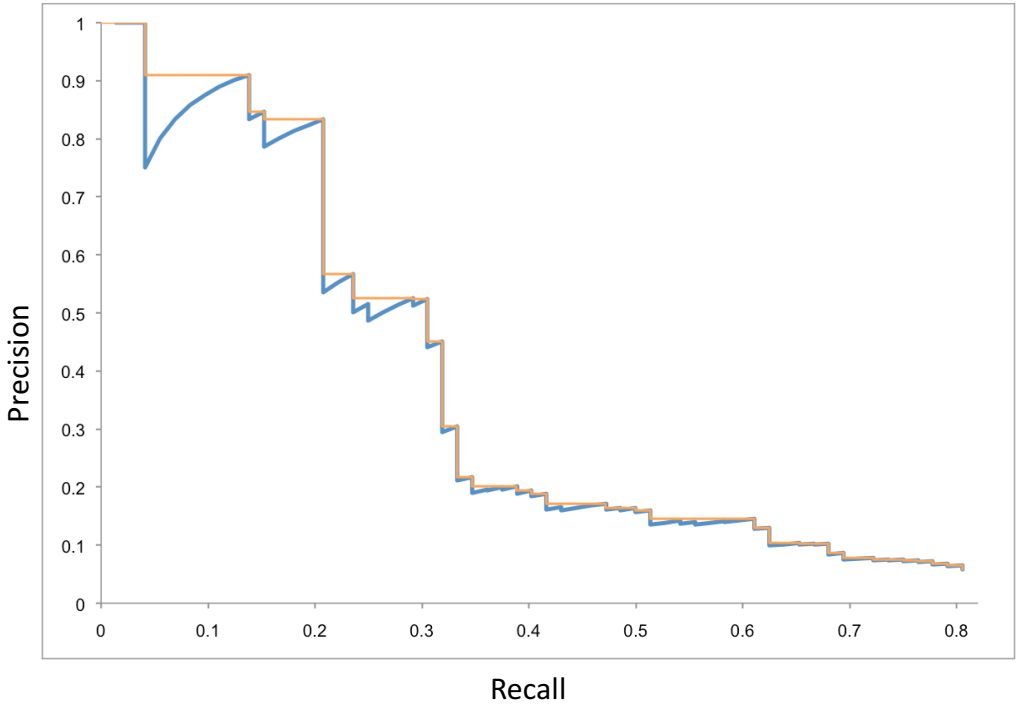
F: Example

	relevant	not relevant	
retrieved	20	40	60
not retrieved	60	1,000,000	1,000,060
	80	1,000,040	1,000,120

- $P = 20/(20 + 40) = 1/3$
- $R = 20/(20 + 60) = 1/4$
- $F_1 = 2\frac{1}{\frac{1}{\frac{1}{3}}+\frac{1}{\frac{1}{4}}} = 2/7$

Evaluating ranked results

Rank	Relevant?	Precision	Recall	Interpolated precision
1	1	1,00	0,01	1,00
2	1	1,00	0,03	1,00
3	1	1,00	0,04	1,00
4	0	0,75	0,04	0,91
5	1	0,80	0,06	0,91
6	1	0,83	0,07	0,91
7	1	0,86	0,08	0,91
8	1	0,88	0,10	0,91
9	1	0,89	0,11	0,91
10	1	0,90	0,13	0,91
11	1	0,91	0,14	0,91
12	0	0,83	0,14	0,85
13	1	0,85	0,15	0,85
14	0	0,79	0,15	0,83
15	1	0,80	0,17	0,83
…	…	…	…	…

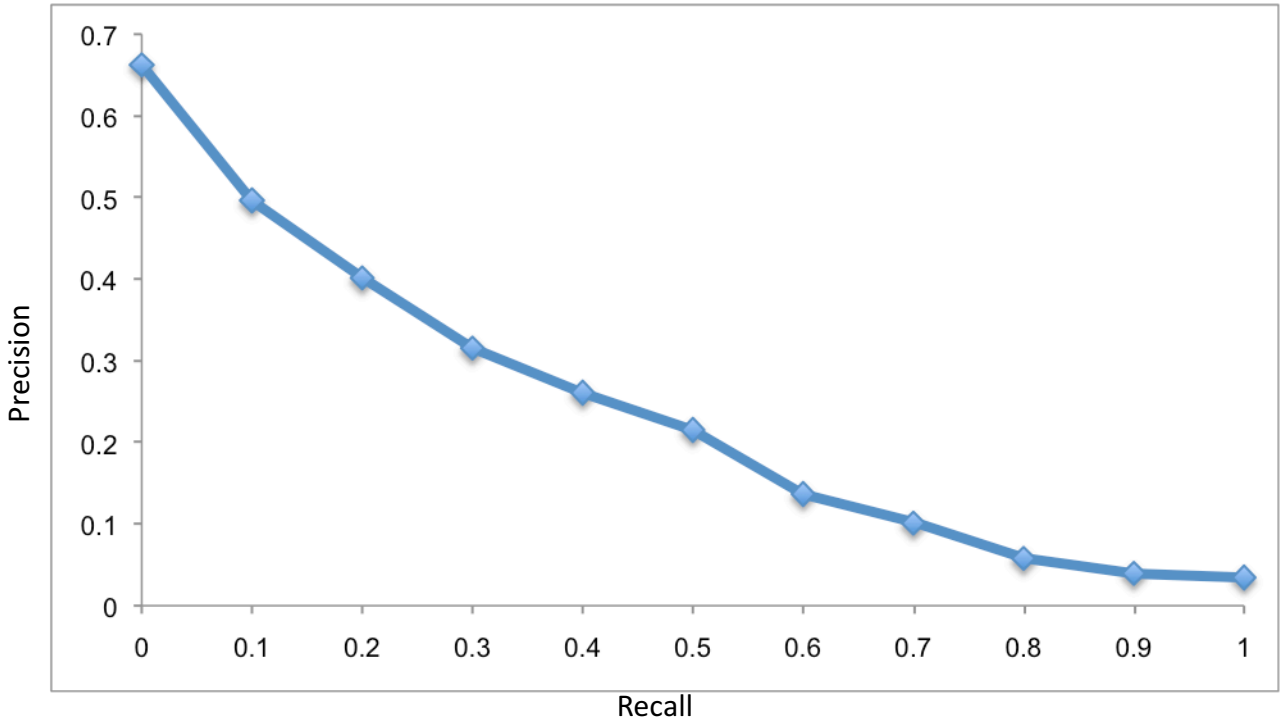


Interpolated precision at recall level *r*:

- maximum precision at any recall level equal or greater than *r*
- Defined for any recall level in [0.0,1.0]

Evaluating ranked results

- Average 11-point interpolated precision for 50 queries



Evaluating ranked results

- Average Precision (AP)
  - average of precision after each relevant document is retrieved
  - Example: AP = 1/1+2/3+3/5+4/7 = 0.7095
  - Mean Average Precision (MAP)

- Precision at K
  - precision after K documents have been retrieved
  - Example: Precision at 10 = 4/10 = 0.4000

- R-Precision
  - For a query with *R* relevant documents, compute precision at R
  - Example: for a query with R=7 relevant docs,

    R-Precision = 4/7 = 0.5714

Rank	Relevant?
1	1
2	0
3	1
4	0
5	1
6	0
7	1
8	0
9	0
10	0

Non-binary relevance

- So far, relevance has been binary
  - a document is either relevant or non-relevant
- The degree to which a document satisfies the user's information need varies
  - Perfect match: *rel = 3*
  - Good match: *rel* = 2
  - Marginal match: *rel = 1*
  - Bad match*:* *rel = 0*

- Evaluate systems assuming that
  - highly relevant documents are more useful than marginally relevant documents, which are more useful than non-relevant ones
  - highly relevant documents are more useful when having higher ranks in the search engine results

Discounted Cumulative Gain

- Cumulative Gain (CG) at rank position *p*

$$CG_p = \sum_{i=1}^{p} rel_i$$

  - Independent of the order of results among the top-*p* positions

- Discounted Cumulative Gain(DCG) at rank position *p*

$$DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i}$$

- Normalized Discounted Cumulative Gain (nDCG) at rank position *p*
  - allows comparison of performance across queries
  - compute ideal DCG_p ($IDCG_p$) from perfect ranking of documents in decreasing order of relevance

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$

Graph of Word – a novel approach for text mining

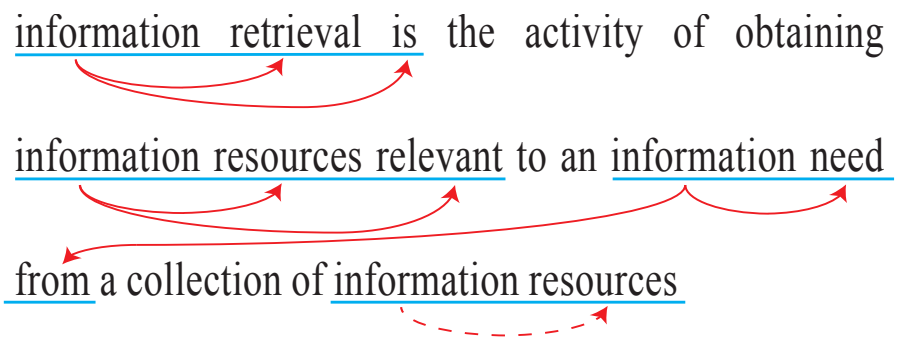
- For efficiency reasons, we used to make the term independence assumption through the bag-of-word representation and the term frequency weighting

- We challenge this assumption by taking into account word dependence and word order through a graph-based representation of a document at indexing time (graph-of-word)

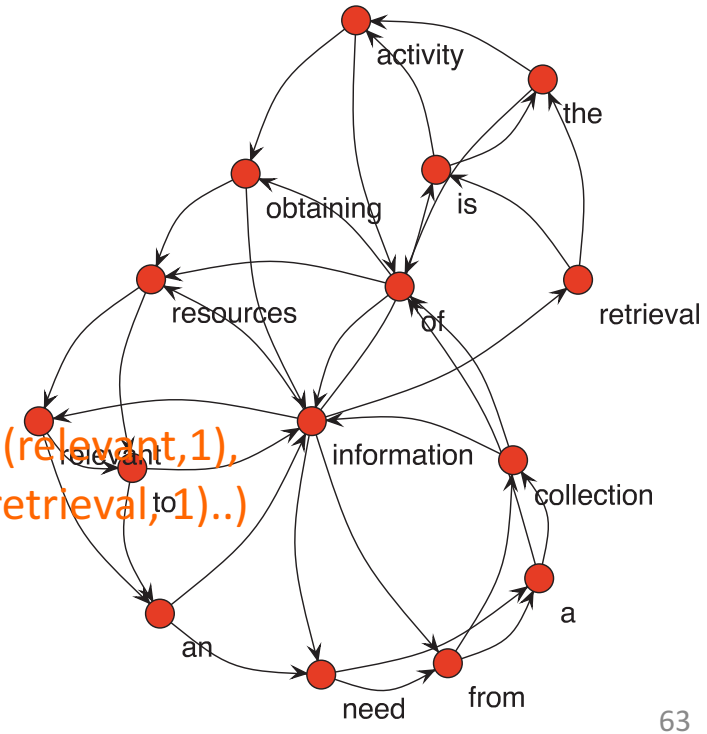
- Graphs have been successfully used in IR to encompass relations between entities and propose meaningful weights (e.g. PageRank[Page et al., 1999])

Graph-based Text Mining

- Instead of the traditional *bag-of-words* (i.e. multiset of terms), we represent a document as a *graph-of-words* to capture *word order* and *word dependency*.



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



Graph of Word – a novel approach for text mining

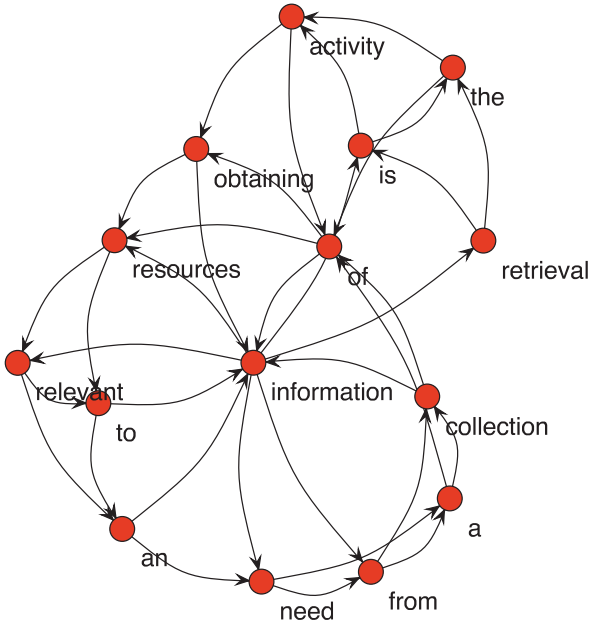
- For efficiency reasons, we used to make the term independence assumption through the bag-of-word representation and the term frequency weighting

- We challenge this assumption by taking into account word dependence and word order through a graph-based representation of a document at indexing time (graph-of-word)

- Graphs have been successfully used in IR to encompass relations between entities and propose meaningful weights (e.g. PageRank[Page et al., 1999])

Graph-based Ad Hoc IR I

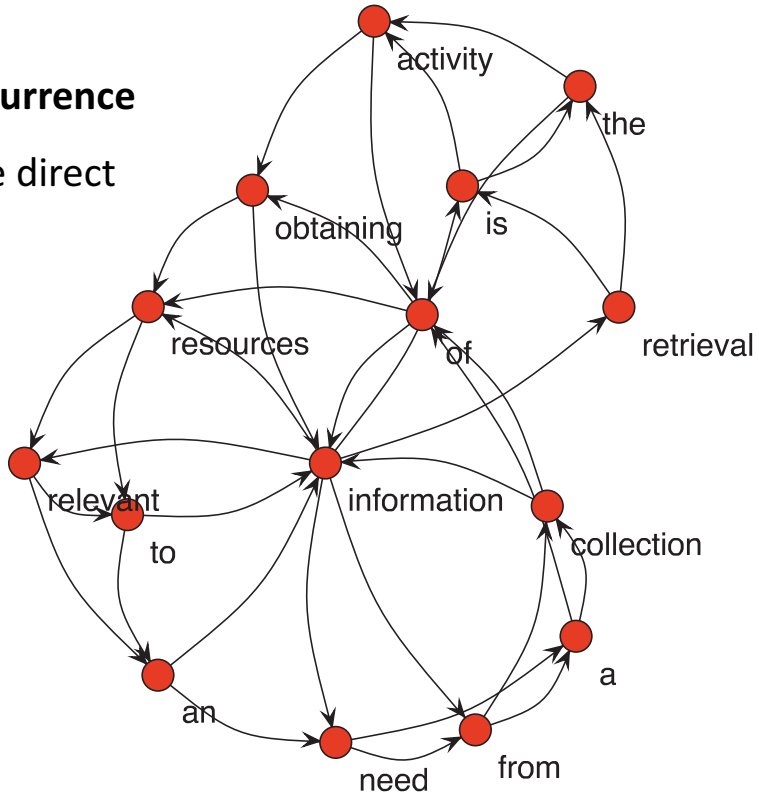
- Ad Hoc Information Retrieval [1,2]
  - Unweighted directed graph
  - The weight of a word in the document: number of neighbors in the graph => favor words that occur with many different other words
  - Robust to varying document length: weight of a word increases only with **new context** of co-occurrences as opposed to the word frequency that increases with any new co-occurrence.



Indegree-BASED TW

- The weight of a term in a document is its **indegree** (numbers of incoming edges) in the **graph-of-word**
- 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



TF-IDF and BM25

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

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

TF-IDF(t, d) = TF_{pol}-IDF(t, d) = $TF_p oTF_l$(t, d) x IDF(t) = $$\left(\frac{1+\log\left(1+\log\left(tf(t,d)\right)\right)}{1-b+b\times\frac{|d|}{avdl}}\right)\times\log\left(\frac{N+1}{df(t)}\right)$$

- **BM25** [Lv and Zhai, CIKM '11]

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

TW-IDF

- Term t, document *d*, collection of size *N*, term weight *tw(t, d)*, document frequency *df(t)*, document length $|d|$, average document length *avdl*, asymptotical marginal gain k_1 (1.2), slope parameter *b*
- 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 our graph-of-word representation, *tw* is the indegree of the vertex representing the term in the graph

EXPERIMENTS

- DATASETS
- PLATFORM
- EVALUATION
- RESULTS

Datasets

- 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 (cont.)

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

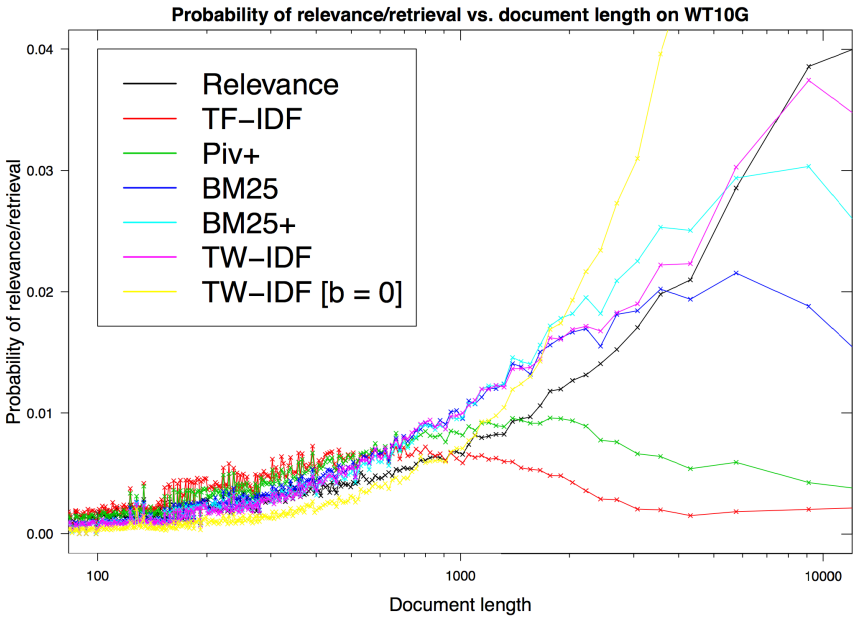
EVALUATION

- 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 ð [Lv and Zhai, CIKM '11]

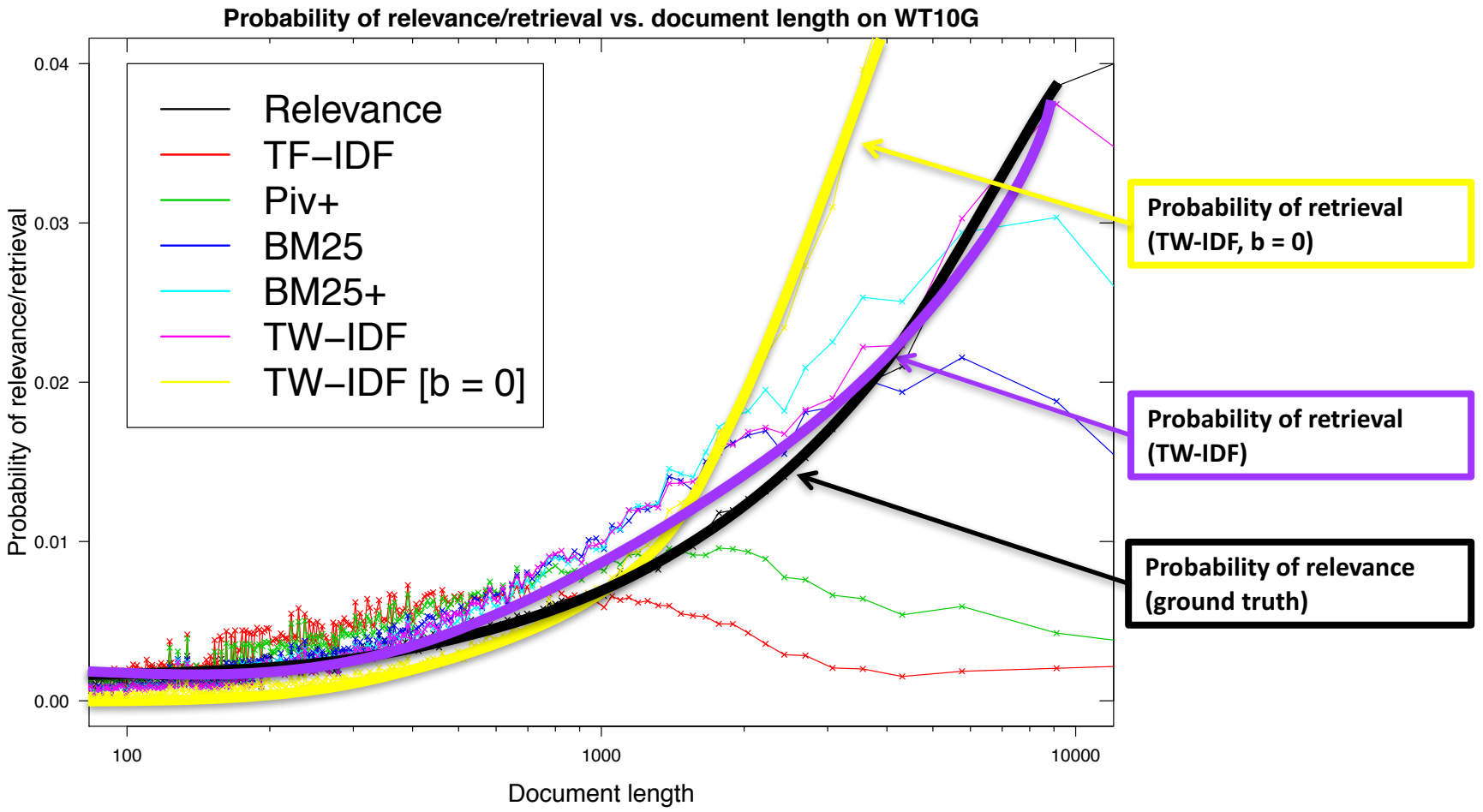
Graph-based Ad Hoc IR II

- 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



Single Document Keyword Extraction [ECIR 2015]

Keywords are used everywhere:

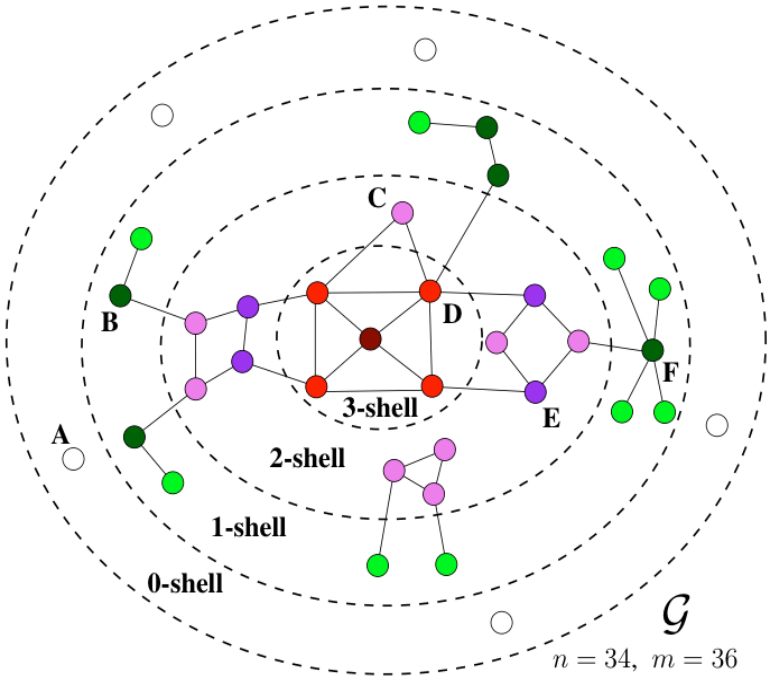
- looking up information on the Web (e. g., via a search engine bar)
- finding similar posts on a blog (e. g., tag cloud)
- for ads matching (e. g., AdWords' keyword planner)
- for research paper indexing and retrieval (e. g., SpringerLink)
- for research paper reviewer assignment

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 degeneracy – k-core

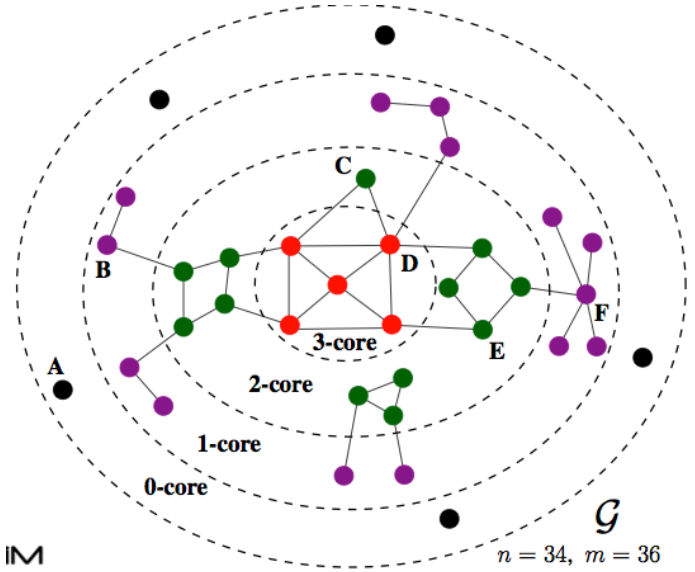
- Let k be an integer.
- A subgraph $H_k = (V', E')$, induced by the subset of vertices $V' \subseteq V$ (and the subset of edges $E' \subseteq E$), is called a *k-core* or *core of order* k iff:
  - $\forall v \in V', degHk(v) \geq k$ and
  - Hk is the maximal subgraph with this property, i.e. it cannot be augmented without losing this property.
- *k-core* of a graph: maximal connected subgraph whose each vertice has at least degree k within the subgraph.
- The core of maximum order is called the *main core*.



Graph-based keyword extraction

Existing graph-based keyword extractors:

- PageRank [Mihalcea and Tarau, 2004]
- HITS [Litvak and Last, 2008]
- assign a *centrality* based influence score to a node
- top ones the most representative



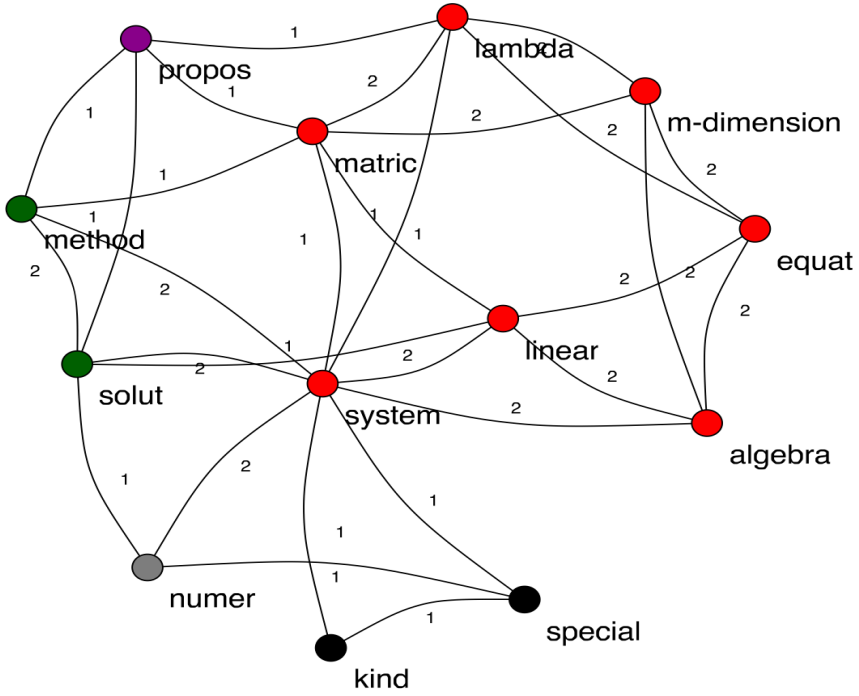
K-core decomposition of the graph

$\Rightarrow$ we propose to retain the main core of the graph to extract the nodes based on their centrality and cohesiveness.

Graph-based Keyword Extraction

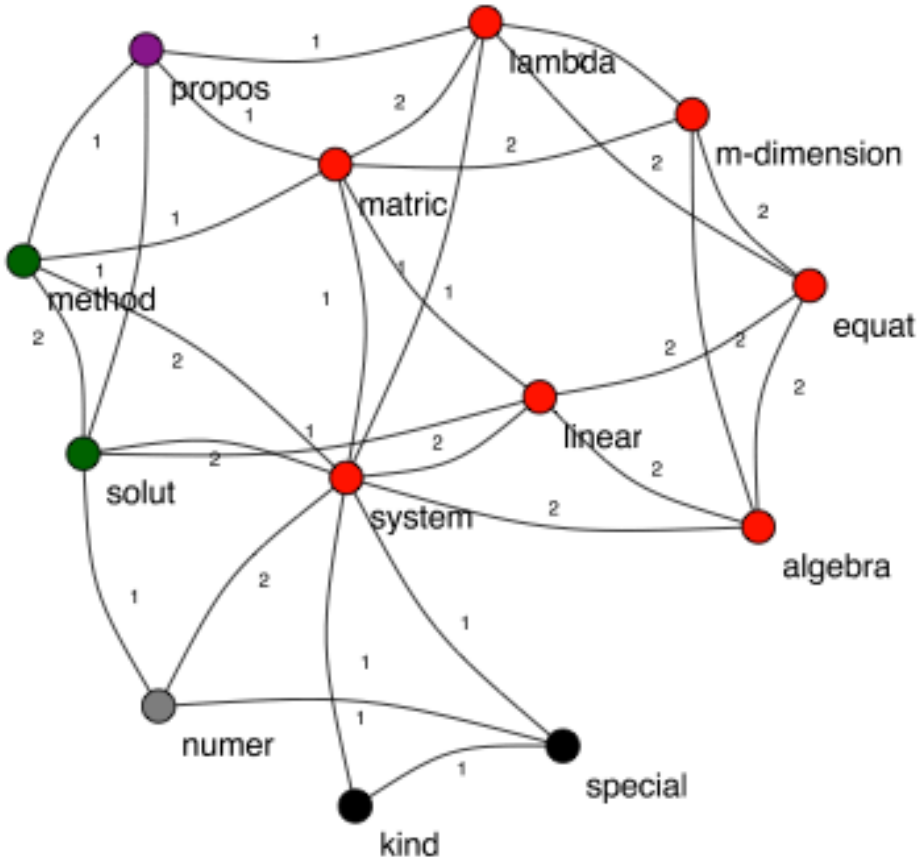
- Single-Document Keyword Extraction
  - Elect the most cohesive sets of words
    in the graph as keywords
  - Use k-core decomposition to extract
    the main core of the graph
  - Weighted edges as opposed to Ad Hoc IR
    (single-document => no normalization)

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. main 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]

K-cores are adaptive

- 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 be decided at document-level (*size of the main core)*.

Data sets

- *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

Graph-of-words:

- undirected edges
- forward edges (natural flow of the text – an edge ***term*1 → *term*2** meaning that ***term*1** precedes ***term*2** in a sliding window)
- backward edges

Keyword extractors:

- PageRank
- HITS (authority scores only)

} Top 33% or top 15% keywords

- K-core
- Weighted K-core

} Main core

Evaluation metrics

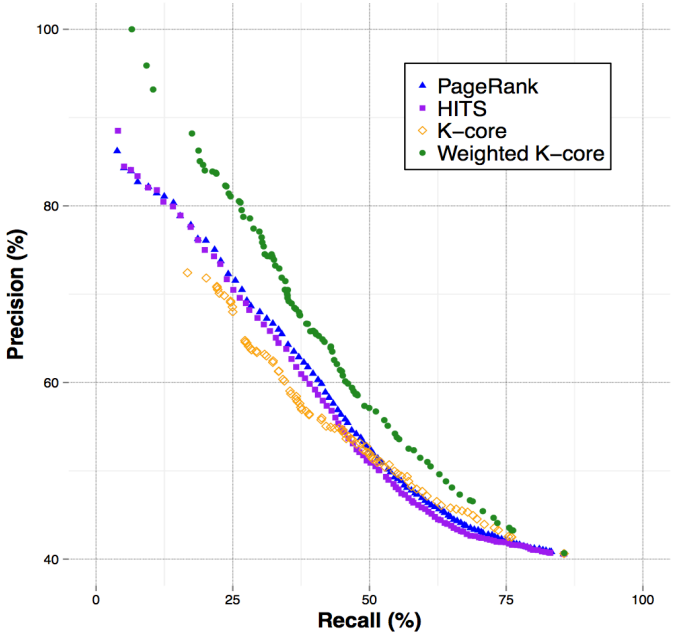
- Each document has a set of golden keywords assigned by humans.
- ⇒ **precision**, **recall** and **F1-score** per document
- ⇒ **macro-average** each metric at the collection level

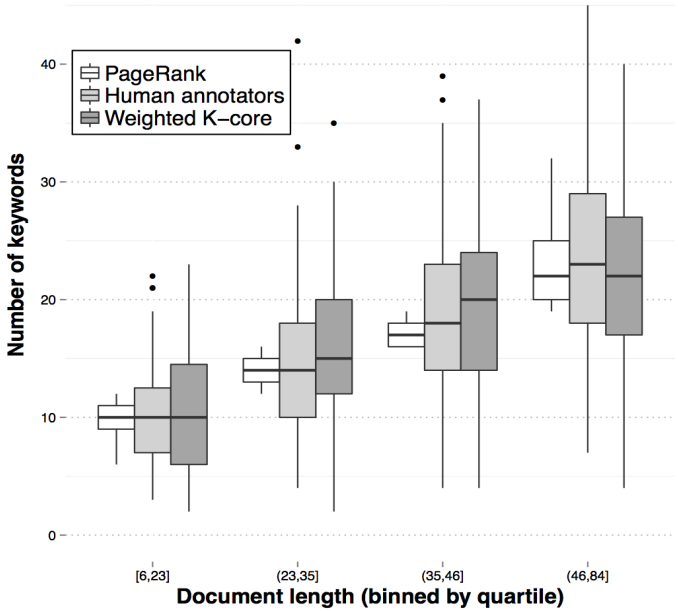
Evaluation metrics 2/2

- Given a golden bigram to extract, how do you penalize a system that retrieves part of it (unigram) or more than it (trigram)?
- ⇒ golden keywords are transformed into unigrams for standard precision and recall definitions.
- ⇒ “reconciliation” step considered as a post-processing step.

Graph-based Keyword Extraction – Experimental results

- Evaluation in terms of:
  - Precision
  - Recall
  - F1-score
  - Precision/recall
  - # of keywords





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

Conclusions – lessons learned

- Explored single-document keyword extraction using k-core on graph-of-words:
  - Capitalized on graph representations of text to extract central terms.
  - Captured more cohesive subgraphs of words central and densely connected.
  - Extracted keywords are more likely to form higher order n-grams and their number adapts to the graph structure.

Eventual example

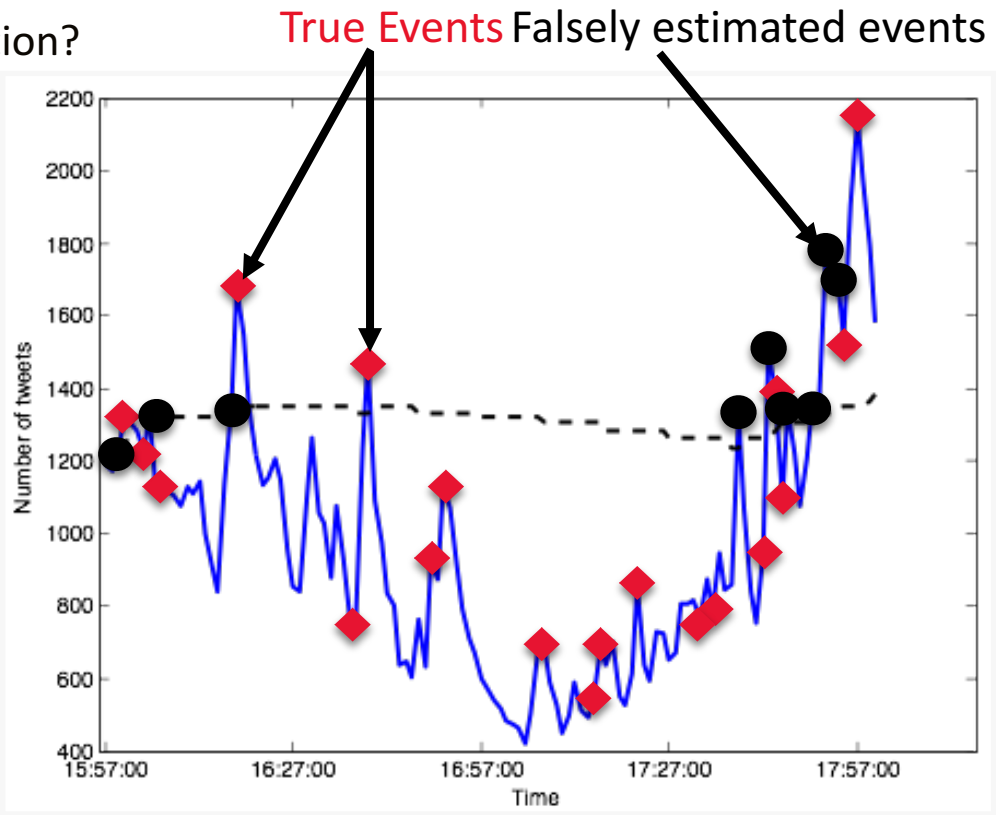
- Stemmed unigrams of the main core of the graph- of-words of the paper document: {*keyword, extract, graph, represent, text, weight, graph-of-word, k-core, degeneraci, edg, vertic, number, document*}.
- Using PageRank, "work" appears in the top 5, "term" and "pagerank" in the top 10, and "case" and "order" in the top 15. Central words but not in cohesion with the rest and probably not relevant.

GoW for text mining

- 1. Ad Hoc Information Retrieval
- 2. Single-Document Keyword Extraction
  - Elect *representative* words for a document that best describe it.
  - Sub-event Detection in Textual Streams (twitter)
  - keyword extraction Chrome extension
- 3. Text Categorization
  - as a Graph Classification Problem
  - Sentence based Kernels for short text categorization

Sub-event Detection in Textual Streams (twitter) [AAAI/ICWSM 2015]

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 histogram of a football match

Contribution

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

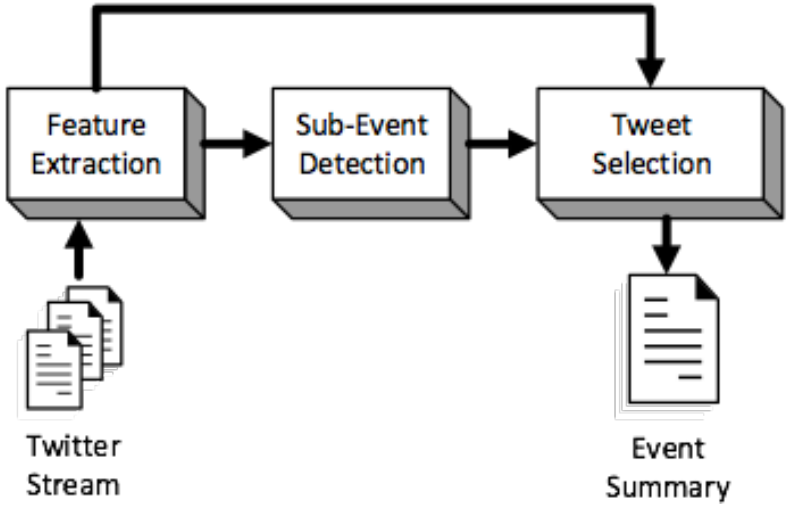
System Architecture

Real Time Event Summarization

1. Feature Extraction: Extracts the terms that best describe the current state of the event

2. Sub-Event Detection: Decides whether a sub-event has occurred

3. Tweet Selection: Ranks all the tweets and selects the first one.



System Architecture

➢Steps are repeated every 60 seconds

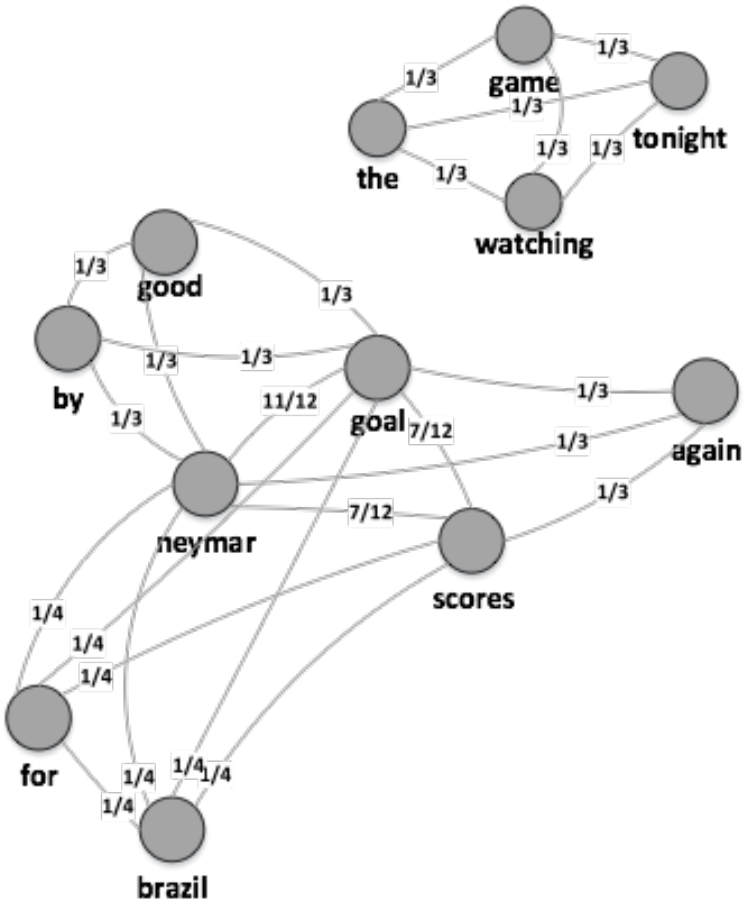
➢The summary of the whole event is constructed by aggregating the individual sub-event descriptions

Graph-of-Words from Twitter Streams

- Represents all the input tweets
  [Rousseau and Vazirgiannis, CIKM 2013]
- node ↔ unique term
- edge ↔ #co-occurrences within a tweet (normalized)

Example Graph:

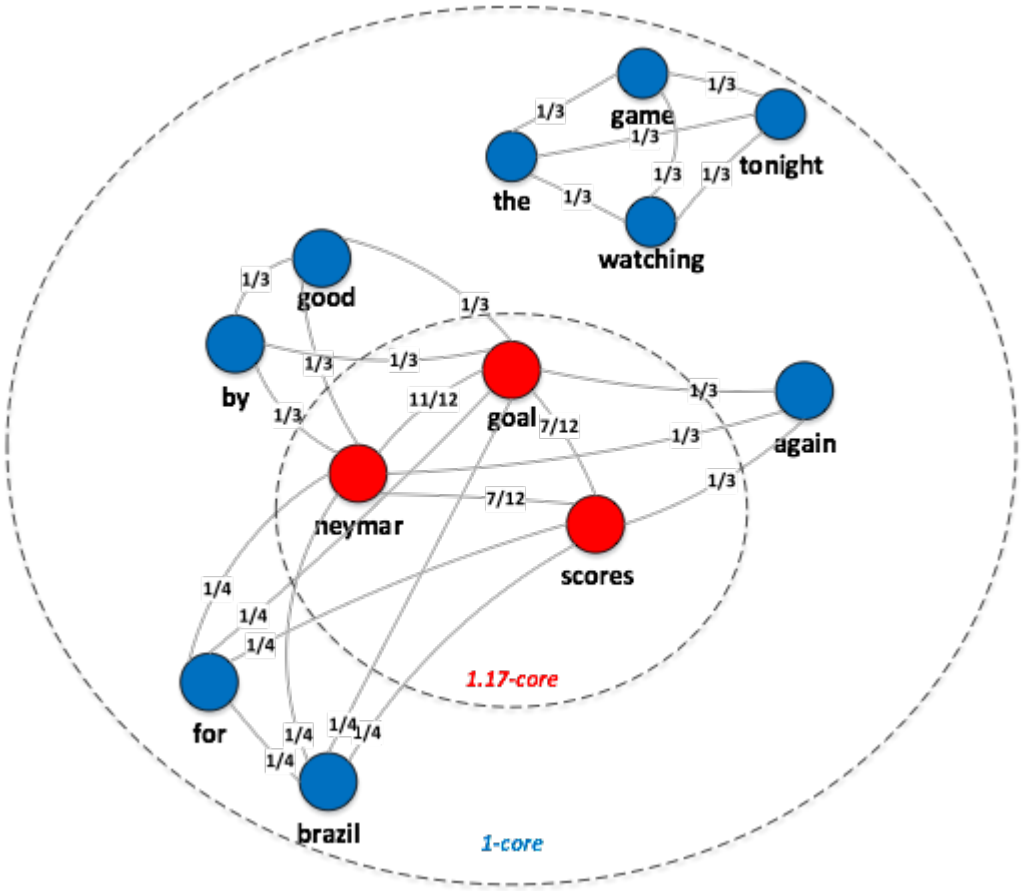
1) Good goal by Neymar
2) Goal! Neymar scores for brazil
3) Goal!! Neymar scores again
4) Watching the game tonight



A graph-of-words built from 4 tweets

Graph Degeneracy for Feature Extraction

- Each term is given a score corresponding to its core number
- The k-core of a graph is the maximal subgraph whose vertices are at least of degree k within the subgraph.
- A vertex has index k if it belongs to the k-core but not to the k+1 core



k-core decomposition of the Graph-of-Words

Sub-event Detection

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

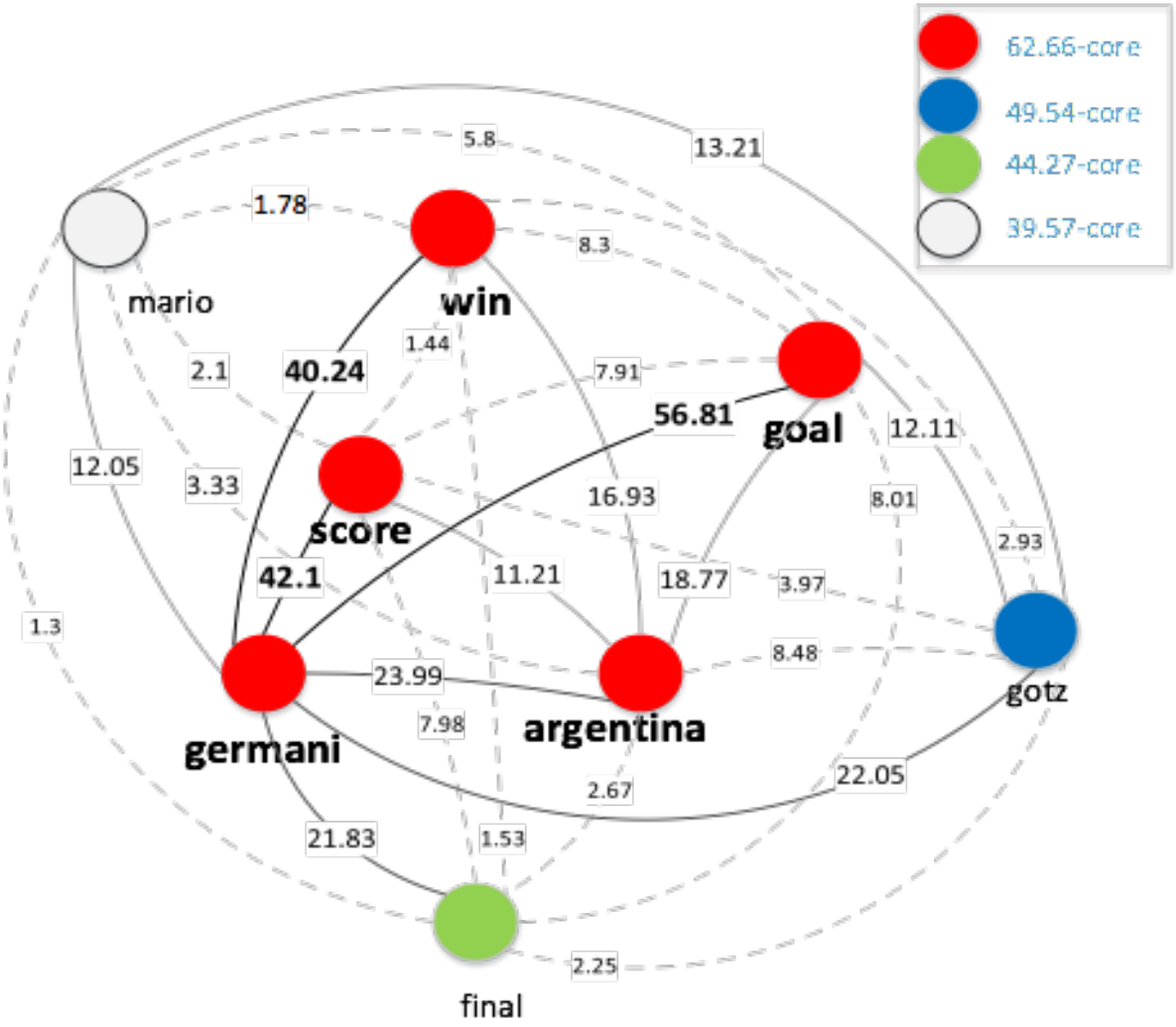
- c_i^t Core number of term i at time slot t
- d Number of terms selected
- θ Decision Threshold
- p Number of previous time slots

Sub-event Detection steps:

(Every 60 seconds)

1) Extract the top d 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: Graph-of-Words

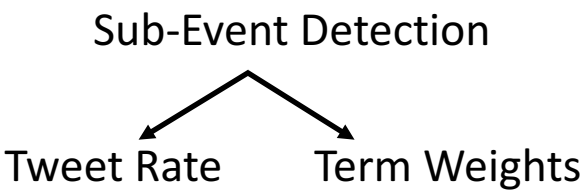


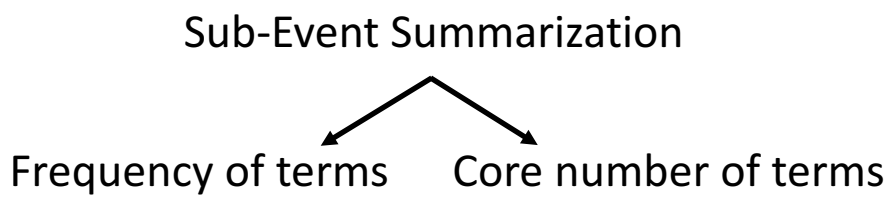
Snapshot of the four highest cores of the graph generated after Germany's goal in the 2014 FIFA World Cup final

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)

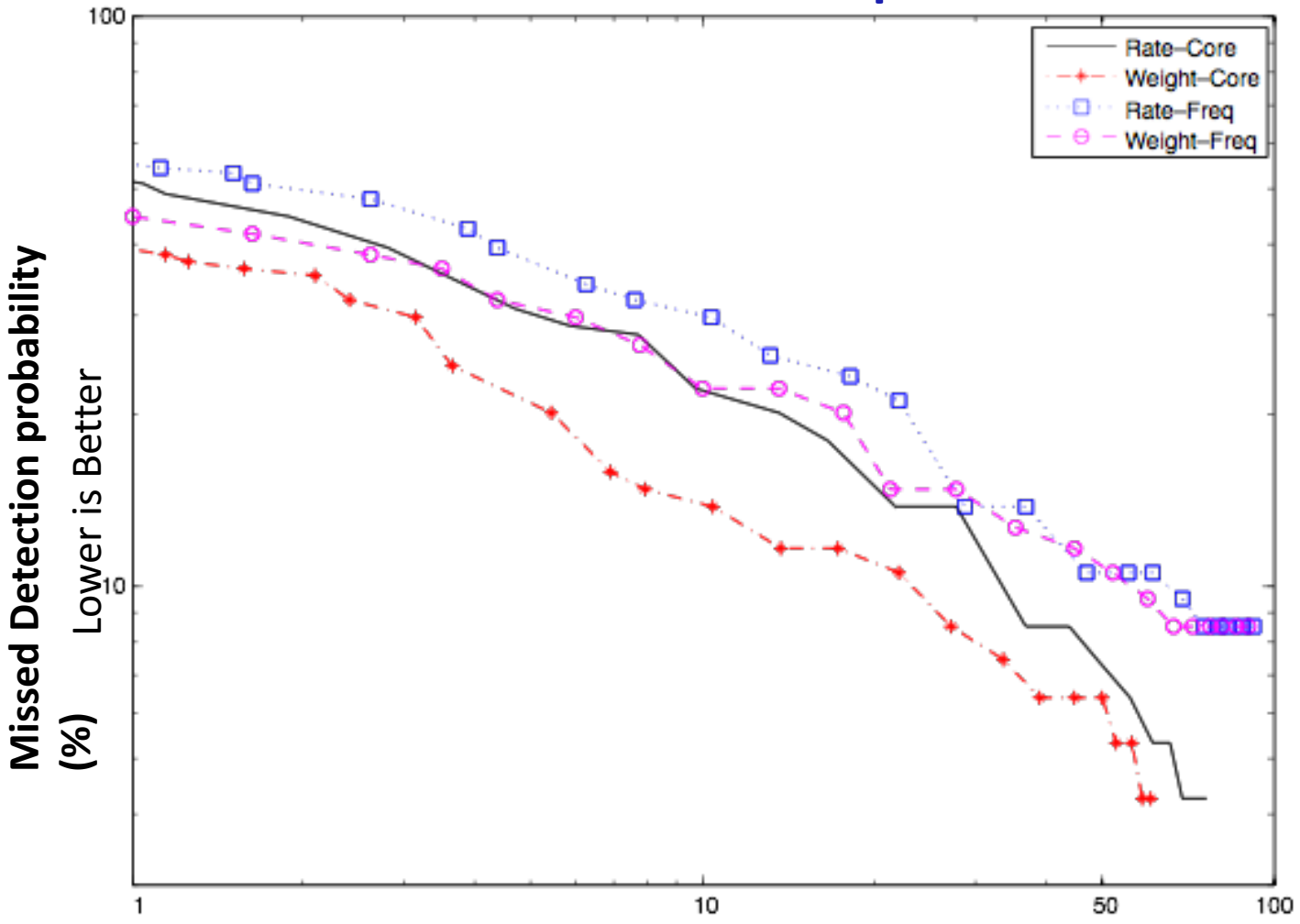
- Rate–Freq: the common baseline
- Rate–Core
- ***Weight–Core***: Our approach
- Weight–Freq

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

Sub-event Detection performance



Average DET curves over 11 matches for the 4 considered approaches.

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	Our 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.

Conclusion

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

GoW for text mining

- 1. Ad Hoc Information Retrieval
- 2. Single-Document Keyword Extraction
  - Elect *representative* words for a document that best describe it.
  - Sub-event Detection in Textual Streams (twitter)
  - keyword extraction Chrome extension
- 3. Text Categorization
  - as a Graph Classification Problem
  - Sentence based Kernels for short text categorization

Chrome addon for keyword extraction

- Based on the main core of the document [ECIR 2015]
- See demo

GoW for text mining

- 1. Ad Hoc Information Retrieval
- 2. Single-Document Keyword Extraction
  - Elect *representative* words for a document that best describe it.
  - Sub-event Detection in Textual Streams (twitter)
  - keyword extraction Chrome extension
- 3. Text Categorization
  - as a Graph Classification Problem
  - Sentence based Kernels for short text categorization

Text Categorization as a Graph Classification Problem **[ACL2015]**

- 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

Context

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

Background

Text categorization [Sebastiani, 2002, Aggarwal and Zhai, 2012]

- 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

- Graph classification
  - subgraphs as features
  - graph kernels [Vishwanathan et al., 2010]
- Frequent subgraph feature mining
  - gSpan [Yan and Han, 2002]
  - FFSM [Huan et al., 2003]
  - Gaston [Nijssen and Kok, 2004]
- expensive to mine all subgraphs, especially for "large" collections of "large" graphs

$\Rightarrow$ 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 ⇒ 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

Standard datasets:

- *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, 2007].
- *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, 2005].
- *LingSpam*: 2,893 emails classified as spam or legitimate messages – split into 10 sets for 10-fold cross validation [Androutsopoulos et al., 2000].
- *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., 2007].

Models

- 3 baseline models (n-gram features)
  - kNN (k=5)
  - Multinomial Naive Bayes (similar results with Bernoulli)
  - linear SVM
- 3 proposed approaches
  - gSpan + SVM (long-distance n-gram features)
  - MC + gSpan + SVM (long-distance n-gram features)
  - MC + SVM (n-gram features)

Evaluation metrics

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

Effectiveness results (1/2)

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

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

Effectiveness results (2/2)

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

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

Dimension reduction – main core

Dataset	# n-grams before	# n-grams after	reduction
WebKB	1,849,848	735,447	60 %
R8	1,604,280	788,465	51 %
LingSpam	2,733,043	1,016,061	63 %
Amazon	583,457	376,664	35 %

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

Unsupervised support selection

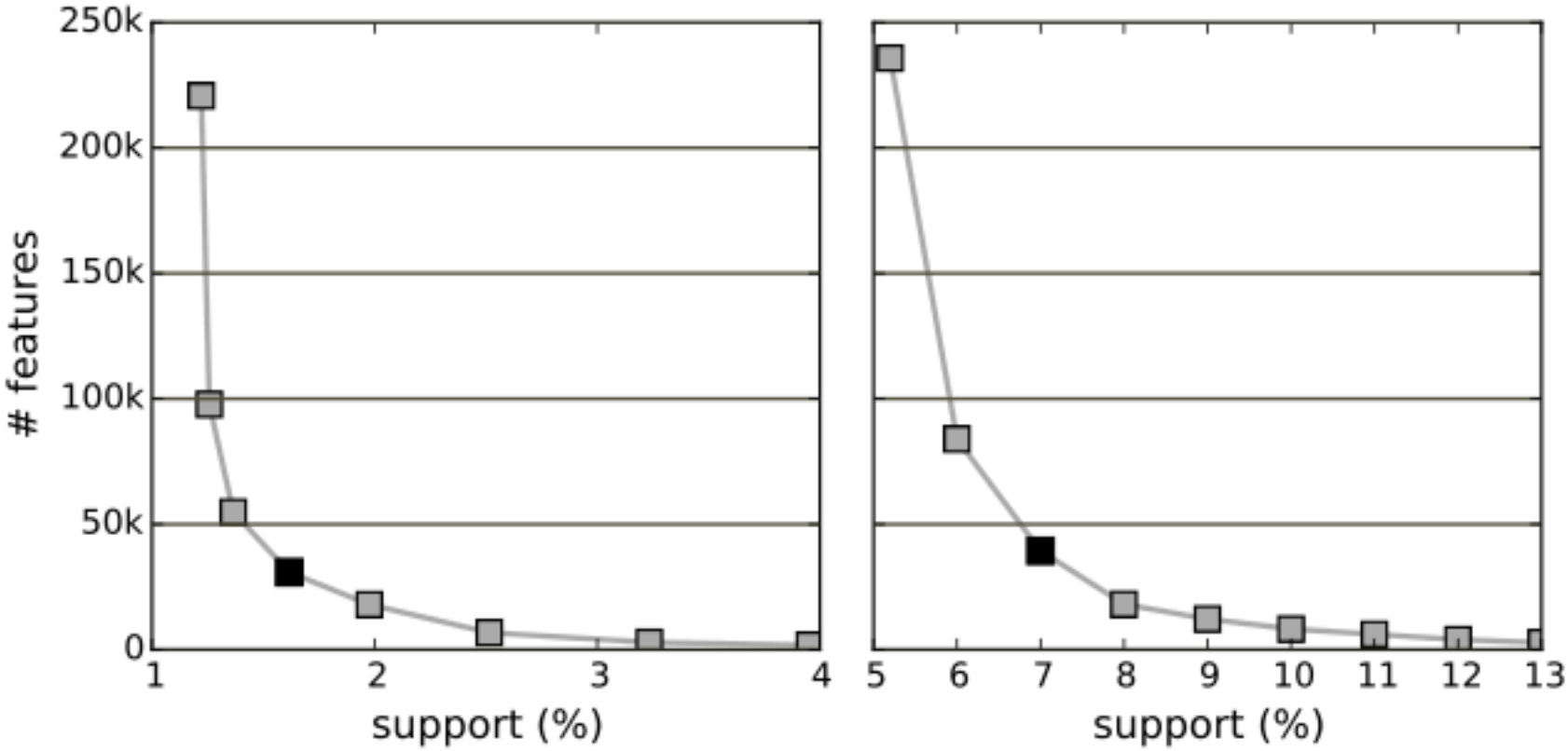


Figure: Number of subgraph features per support (%) on WebKB (left) and R8 (right) datasets. In black, the selected support chosen via the elbow method.

Distribution of mined n-grams

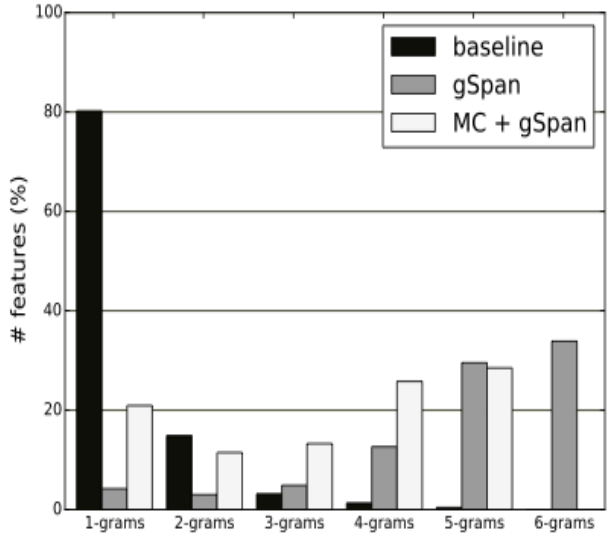


Figure: Distribution of n-grams (standard and long-distance ones) among all the features on WebKB dataset.

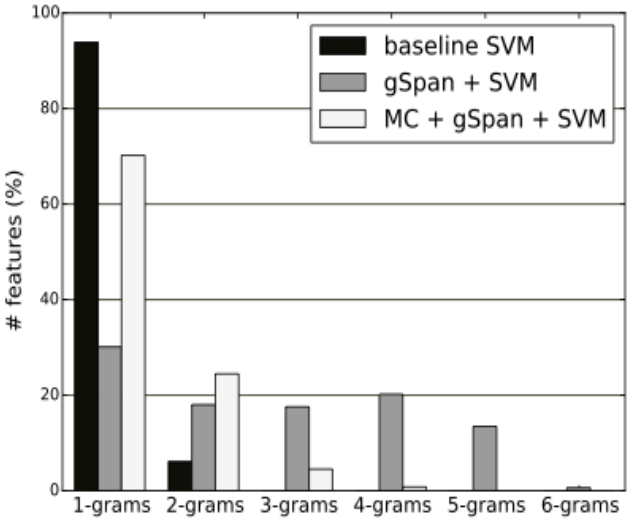


Figure: Distribution of n-grams (standard and long-distance ones) among the top 5% most discriminative features for SVM on WebKB dataset.

Contribution

- we explored a graph-of-words, to challenge the traditional bag-of-words for text classification.
- first trained a classifier using frequent sub- graphs 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.

Graph of Words - Contributions

- Extraction – GoW for term weighting for IR, keyword extraction and classification
- Learning @ document level
  - new features - subgraphs of word for classification
- Regularization @ collection level
  - GoW - regularization as feature similarity (similar features should have similar weights

Future directions

- GoW as features to ML algorithms
- GoW for Document and Collection visualization
- Move from tw-idf to tw-icw and tw-idw …
- designing new kernels for syntactic parse trees – similarity between non-terminal nodes
- graph kernels for document similarity
- graph based regularization for text classification
- gowis - acl 2016
- keyword extraction for summarization - off line
- keyword extraction for summarization - on line

Relevant publications

① Rousseau, F. and M. Vazirgiannis (2013a). Composition of TF Normalizations: New Insights on Scoring Functions for Ad Hoc IR. In *Proceedings of ACM SIGIR '13, pp. 917–920*.

② Rousseau, F. and M. Vazirgiannis (2013b). Graph-of-word and TW-IDF: New Approach to Ad Hoc IR. In *Proceedings of the 22nd ACM CIKM '13*, pp. 59–68, *best paper mention award*.

③ Rousseau, F. and M. Vazirgiannis (2015a). Main Core Retention on Graph-of-words for Single-Document Keyword Extraction. *In Proceedings of the 37th European Conference on Information Retrieval. ECIR '15*.

④ P. Meladianos, Y. Nikolentzos, F. Rousseau, Y. Stavrakas and M. Vazirgiannis (2015b)Degeneracy-based Real-Time Sub-Event Detection in Twitter Stream. Proceedings of the *9th AAAI International Conference on Web and Social Media (AAAI ICWSM '15).*

⑤ Rousseau, F., Kiagias E.and M. Vazirgiannis, (2015c) Text Categorization as a Graph Classification Problem, in the proceedings of the *ACL 2015* conference

⑥ Jonghoon Kim F. Rousseau M. Vazirgiannis, "Convolutional Sentence Kernel from Word Embeddings for Short Text Categorization", EMNLP 2015 conference

⑦ Konstantinos Skianis, François Rousseau, Michalis Vazirgiannis:Regularizing Text Categorization with Clusters of Words. EMNLP 2016: 1827-1837

⑧ Antoine J.-P. Tixier, Fragkiskos D. Malliaros, Michalis Vazirgiannis:A Graph Degeneracy-based Approach to Keyword Extraction. EMNLP 2016: 1860-1870

⑨ GoWvis: a web application for Graph-of-Words-based text visualization and summarizationAJP Tixier, K Skianis, M VazirgiannisACL 2016, 151

References

- Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. http://www-nlp.stanford.edu/IR-book/
- “Indexing by Latent Semantic Analysis”, S.Deerwester, S.Dumais, T.Landauer, G.Fumas, R.Harshman, Journal of the Society for Information Science, 1990
- “Mining the Web: Discovering Knowledge from Hypertext Data”, Soumen Chakrabarti