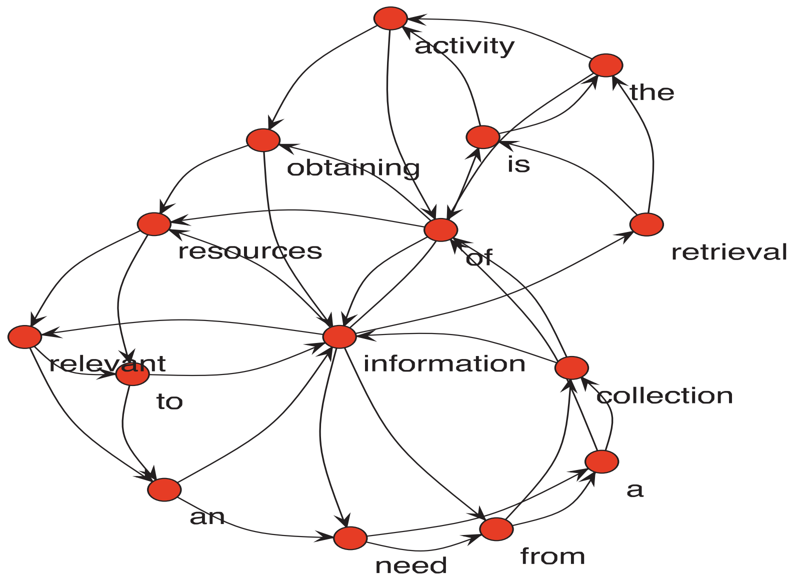
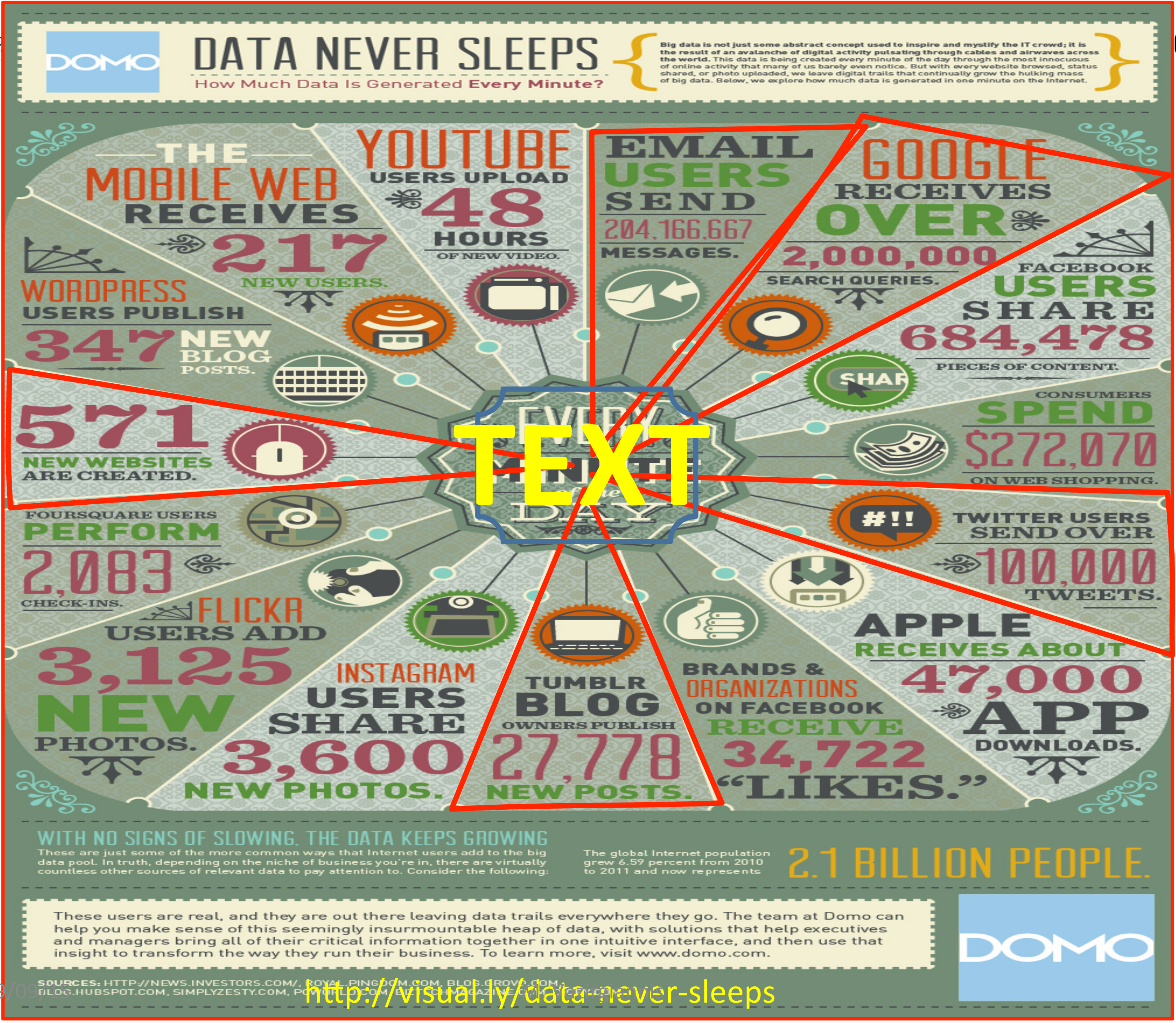
# Graph-of-words: boosting text mining with graphs



M. Vazirgiannis

Data Science & Mining group
LIX @ Ecole Polytechnique

http://www.lix.polytechnique.fr/dascim/



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.

CONSUMERS SPEND $272,070 ON WEB SHOPPING.

TWITTER USERS SEND OVER 100,000 TWEETS.

APPLE RECEIVES ABOUT 47,000 APP DOWNLOADS.

BRANDS & ORGANIZATIONS ON FACEBOOK RECEIVE 34,722 "LIKES."

TUMBLR BLOG OWNERS PUBLISH 27,778 NEW POSTS.

INSTAGRAM USERS SHARE 3,600 NEW PHOTOS.

FLICKR USERS ADD 3,125 NEW PHOTOS.

FOURSQUARE USERS PERFORM 2,083 CHECK-INS.

571 NEW WEBSITES ARE CREATED.

WORDPRESS USERS PUBLISH 347 NEW BLOG POSTS.

EVERY MINUTE of the DAY

TEXT

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/, ROYAL.PINGDOM.COM, BLOG.GROVO.COM, BLOG.HUBSPOT.COM, SIMPLYZESTY.COM, POWERFLOW.COM, BIZTECHMAGAZINE.COM, [illegible]

http://visual.ly/data-never-sleeps

# Text Mining - Where?

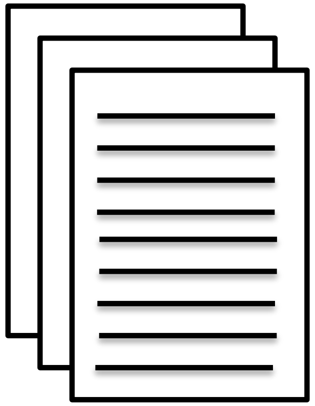
- Web search engines
  - Understand the user query (i.e. Jaguar…)
  - Find matching documents (ranking)
  - Group similar results together (clustering)
  - etc.
- Product recommendation
  - Understand product descriptions
  - Understand product reviews
  - etc.

## Text Mining

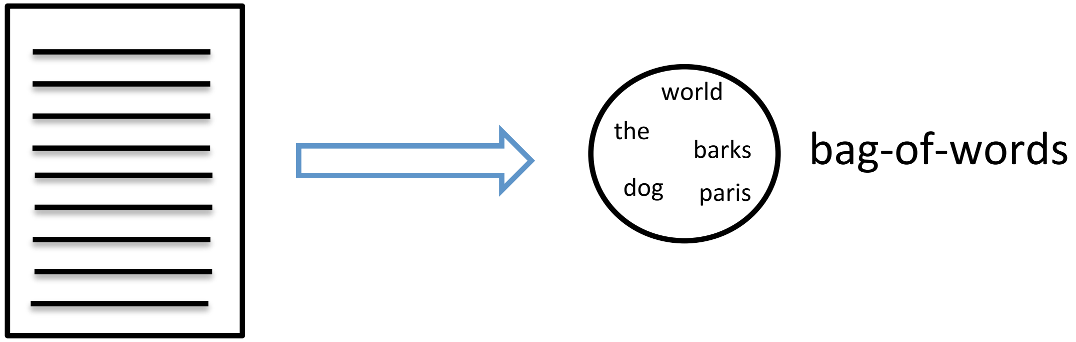
- Analogy with the petroleum industry: *Explore, extract, refine & use*
- Here *the oil is the information* contained in textual documents
- Text mining involves:
  - ➢Data Mining & Machine Learning (ML)
  - ➢Information Retrieval (IR)
  - ➢Natural Language Processing (NLP)

# Text Mining – Terminology

- You apply text mining on a collection of documents:
  - ➢the collection is the data set
  - ➢the documents are the data points

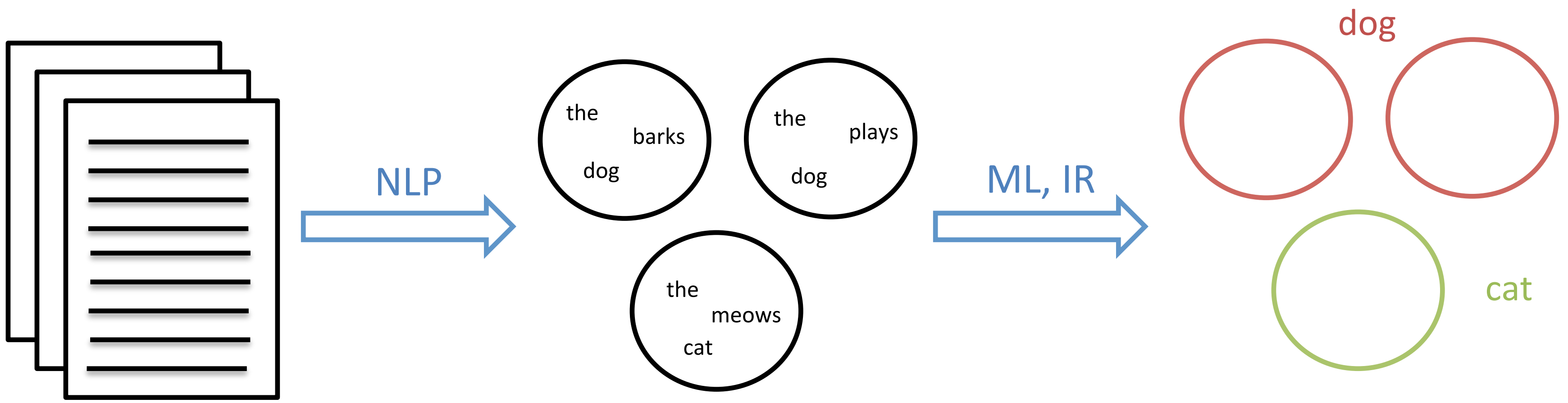


- Because text is unstructured, a document is usually converted in a common *representation*

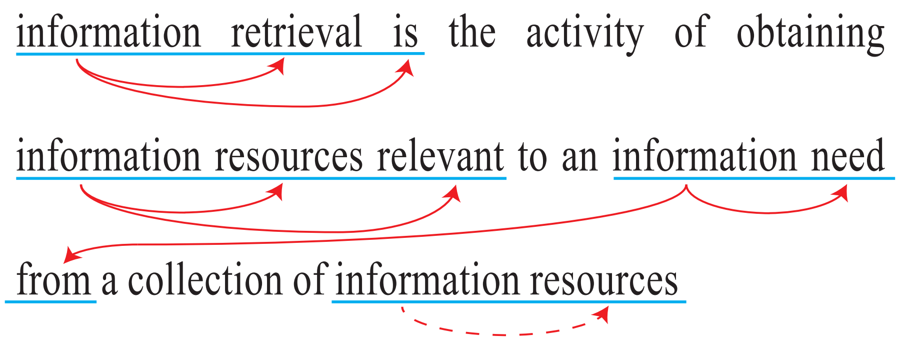


# Text Mining – Pipeline

- Input: raw text
- Output: some information (application-dependent)
- In-between:
  1) document representation
  2) feature extraction – bag of words



# Bag of Words - issues


information retrieval is the activity of obtaining
information resources relevant to an information need
from a collection of information resources

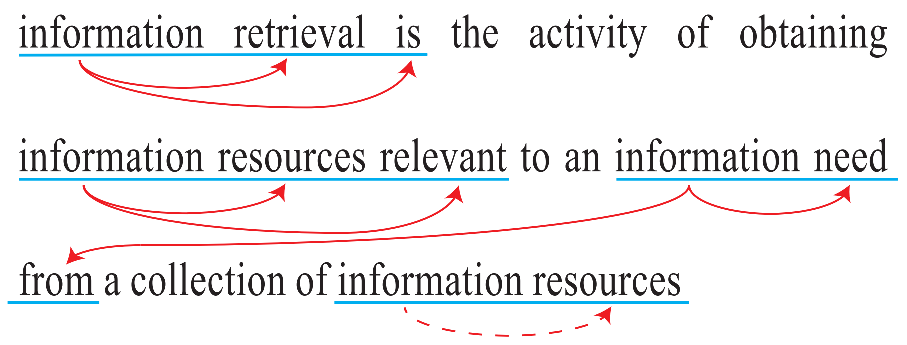
- term independence assumption
- term frequency weighting

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

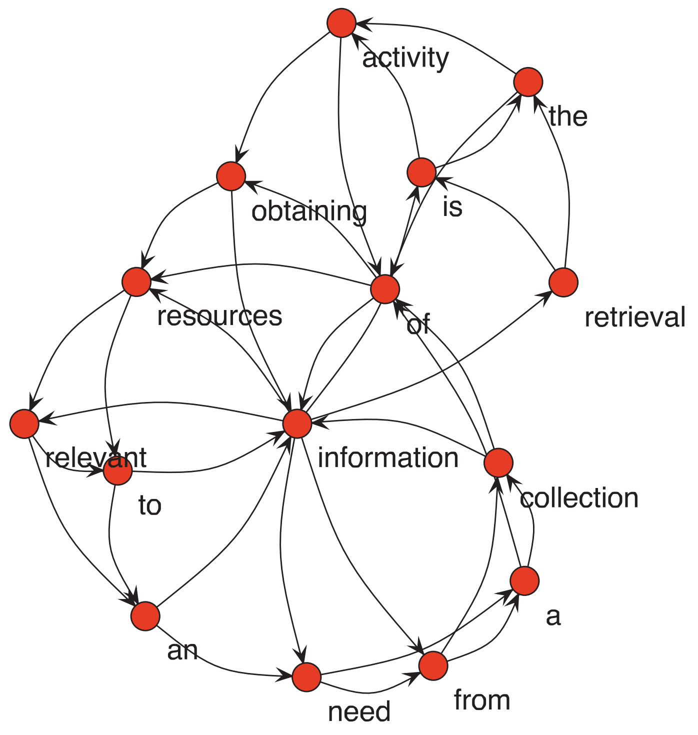
# Graph of Words – a novel approach for text mining

- Challenge "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  successfully used in IR to encompass relations and propose meaningful weights (e.g. PageRank)

# Graph of Words (GoW)


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)..)



Captures: frequency, order and distance, …

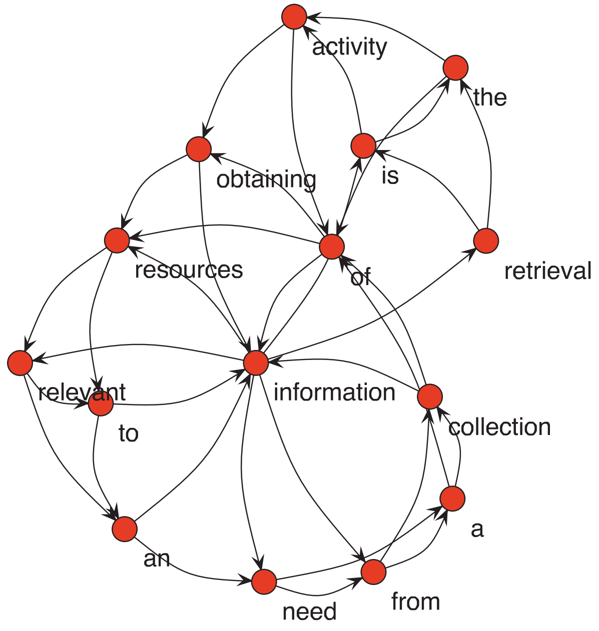
## GoW for text mining

- 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

# Graph-based Ad Hoc IR [CIDKM2013]

Ad Hoc Information Retrieval [1,2]

- Unweighted directed graph
- word weighting in the document: number of neighbors in the graph
  - => favor words that co-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.



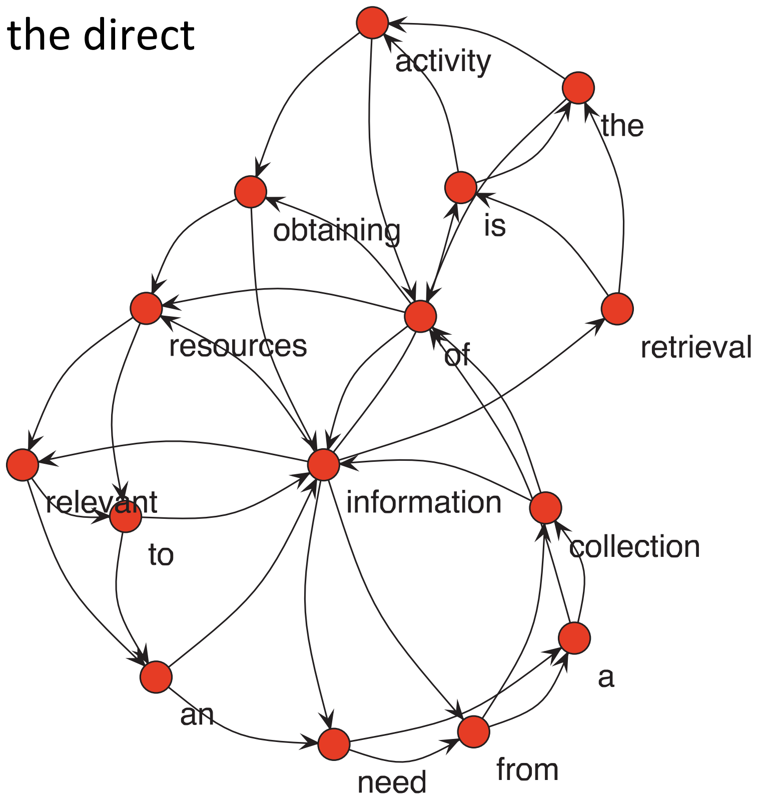
# Other efforts involving graphs

- Graph representations of text [Blanco and Lioma, 2012]:
  - To challenge the traditional bag-of-words document representation
- take into account some term dependence and term order
- node as  linguistic unit (i.e., sentence, word or a character)
- edge as linguistic relationship (syntactic or semantic)
- syntactic graph-of-words with edges representing co-occurrences in a small sliding window.

# 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
- 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]

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

## BM25 [Lv and Zhai, CIKM '11]

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

# TW-IDF

- *tw*: indegree of the vertex representing the term in the graph

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

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

# Experiments - 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 |

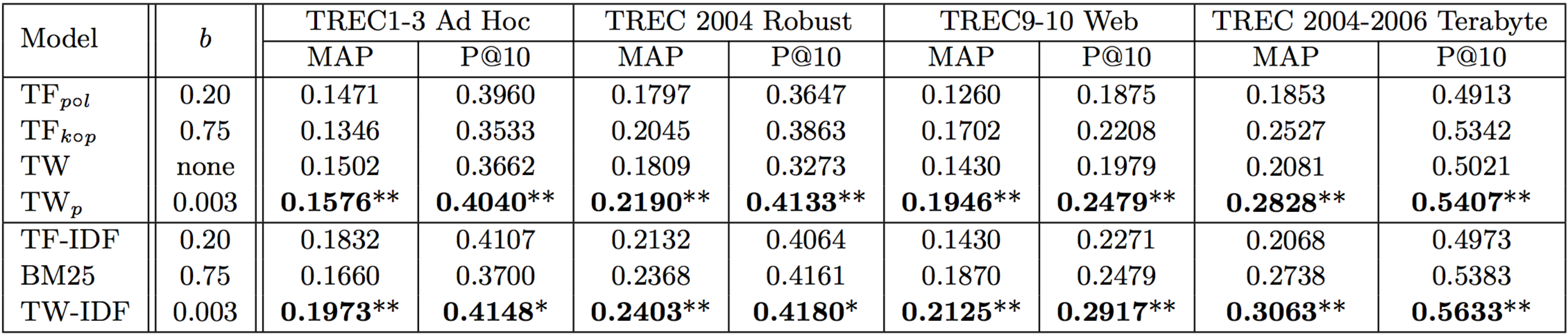
**Table 1**: Statistics on the four TREC datasets used; Disks 4&5 excludes the Congressional Record. The average values are computed per document.

# 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
  - 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+

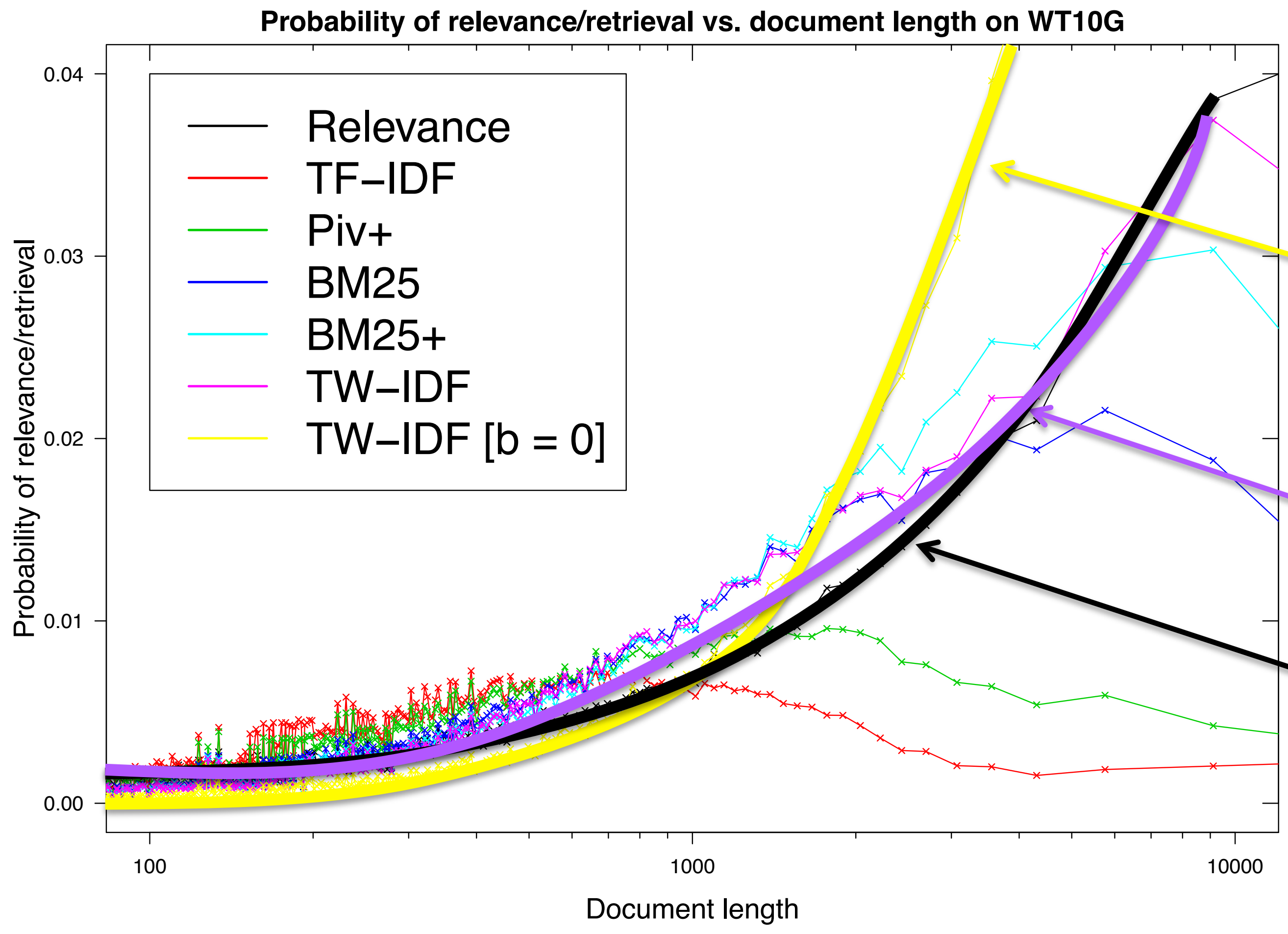
# Graph-based Ad Hoc IR II

- 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 |
| $\text{TF}_{pol}$ | 0.20 | 0.1471 | 0.3960 | 0.1797 | 0.3647 | 0.1260 | 0.1875 | 0.1853 | 0.4913 |
| $\text{TF}_{kop}$ | 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 |
| $\text{TW}_{p}$ | 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



**Probability of retrieval (TW-IDF, b = 0)**

**Probability of retrieval (TW-IDF)**

**Probability of relevance (ground truth)**

## 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

# Single Document Keyword Extraction [ECIR 2015]

Keywords are used everywhere:

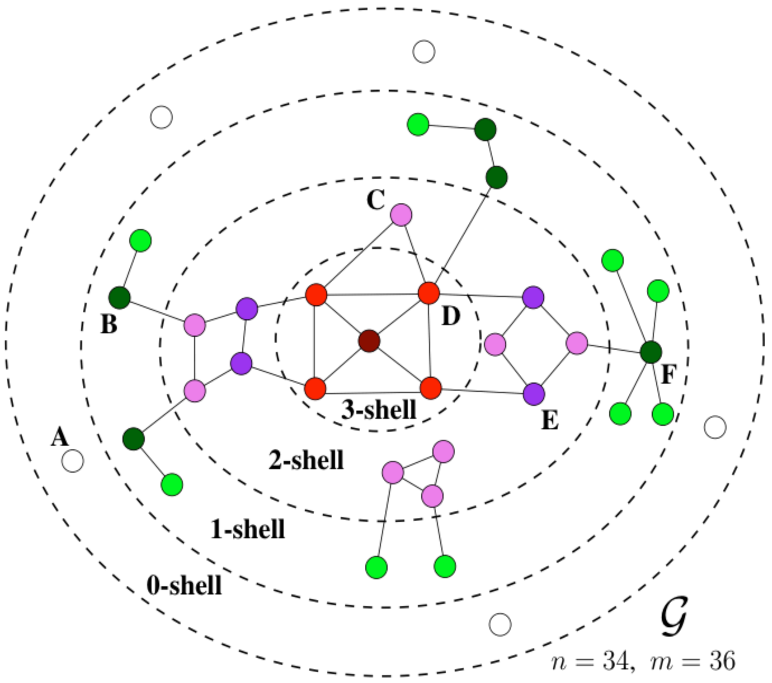
- looking up information on the Web (e. g., via a search engine bar)
- finding similar posts on a blog (e. g., tag cloud)
- for ads matching (e. g., AdWords' keyword planner)
- for research paper indexing and retrieval (e. g., SpringerLink)
- for research paper reviewer assignment

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:
  - $\forall v \in V'$, $deg_{Hk}(v) \geq k$ 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*.



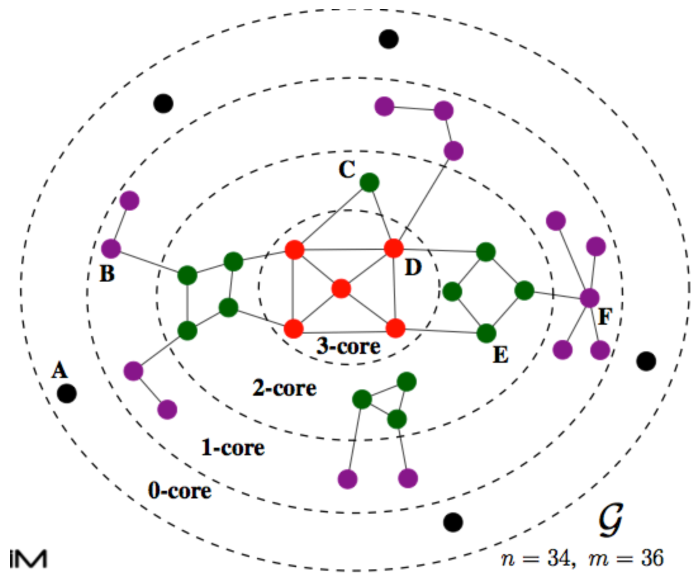
# Graph-based keyword extraction

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

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

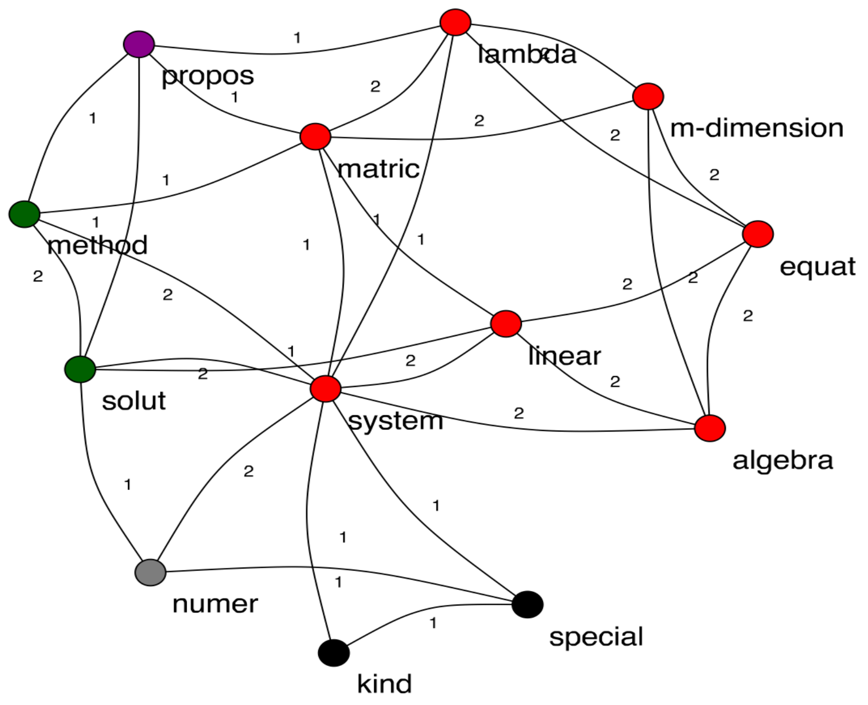


K-core decomposition of the graph

# Graph-based Keyword Extraction

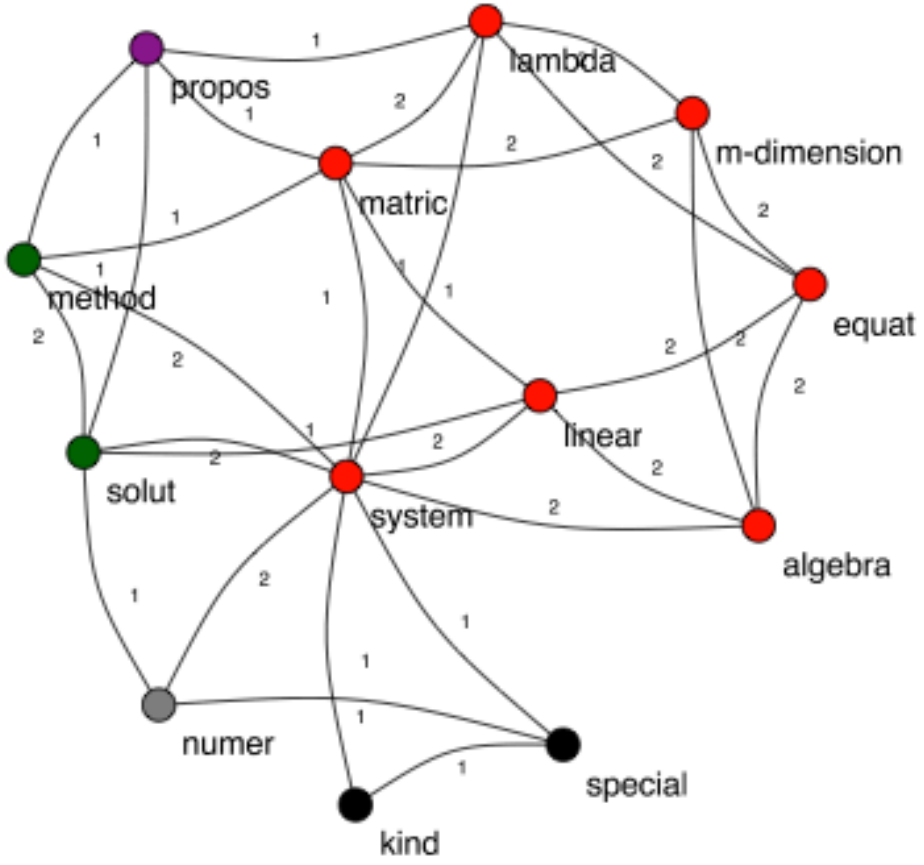
- Single-Document Keyword Extraction
  - ➢ Elect the most cohesive sets of words in the graph as keywords
  - ➢ Use k-core decomposition to extract the main core of the graph
  - ➢ Weighted edges as opposed to Ad Hoc IR (single-document => no normalization)

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)
- K-core
- Weighted K-core

$\left.\begin{matrix}\text{PageRank}\\\text{HITS}\end{matrix}\right\}$ Top 33% or top 15% keywords

$\left.\begin{matrix}\text{K-core}\\\text{Weighted K-core}\end{matrix}\right\}$ Main core

## Evaluation metrics

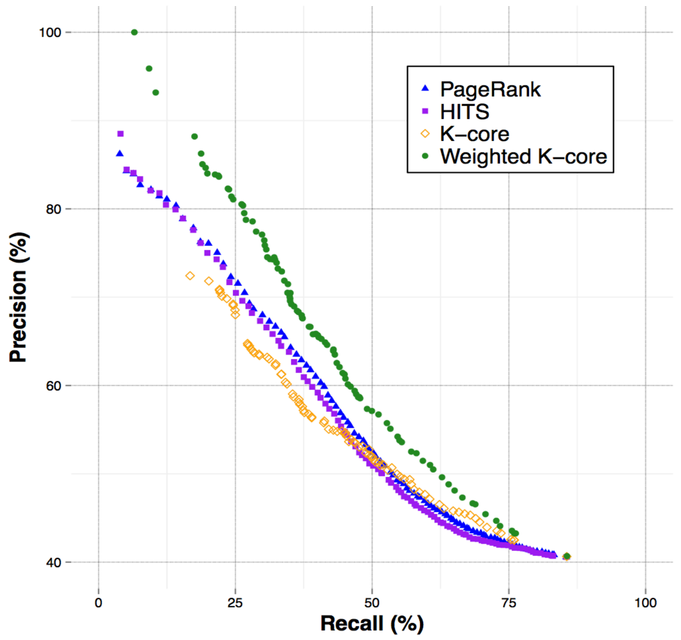
- Each document has a set of golden keywords assigned by humans.
- ⇒ **precision**, **recall** and **F1-score** per document
- ⇒ **macro-average** each metric at the collection level

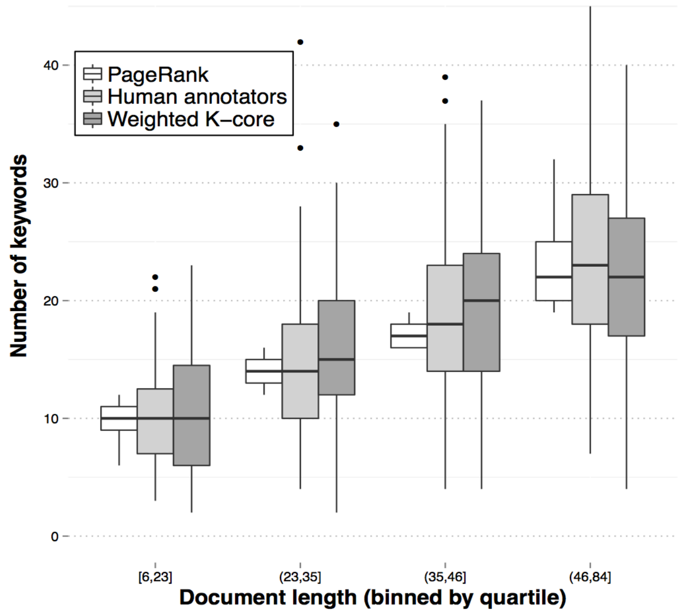
# Evaluation metrics 2/2

- 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

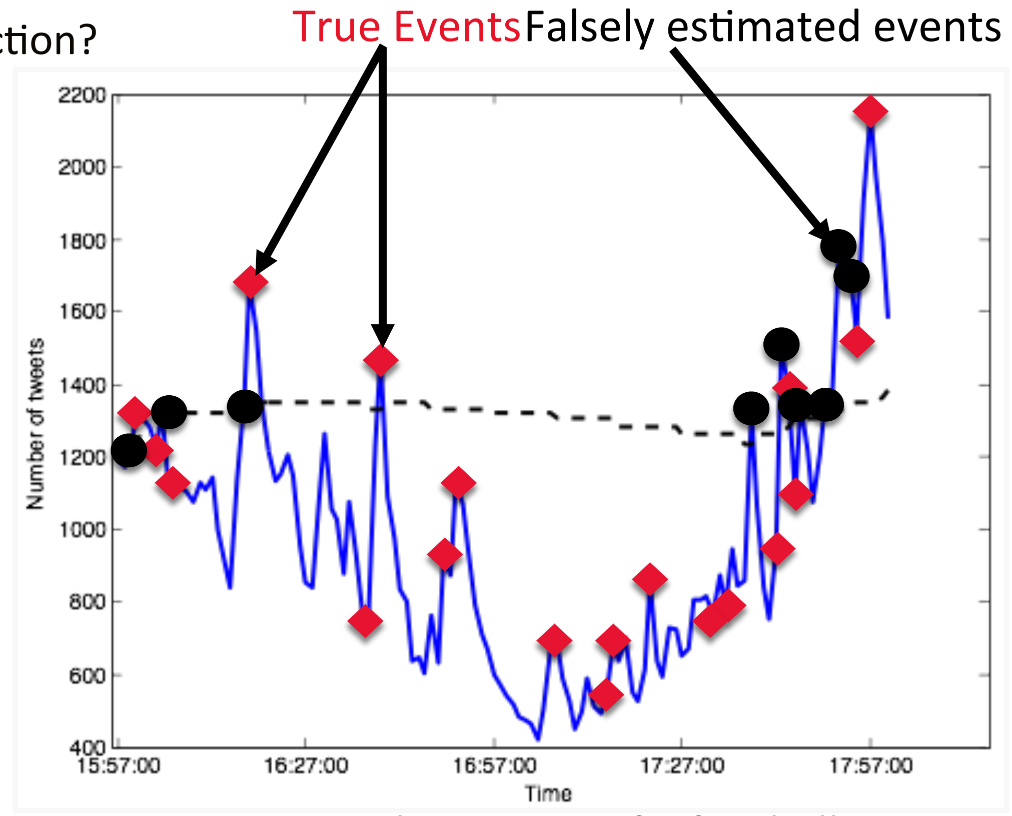
- Stemmed unigrams of the main core of the graph- of-words of the paper document: {*keyword, extract, graph, represent, text, weight, graph-of-word, k-core, degeneraci, edg, vertic, number, document*}.
- Using PageRank, “work” appears in the top 5, “term” and “pagerank” in the top 10, and “case” and “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.*

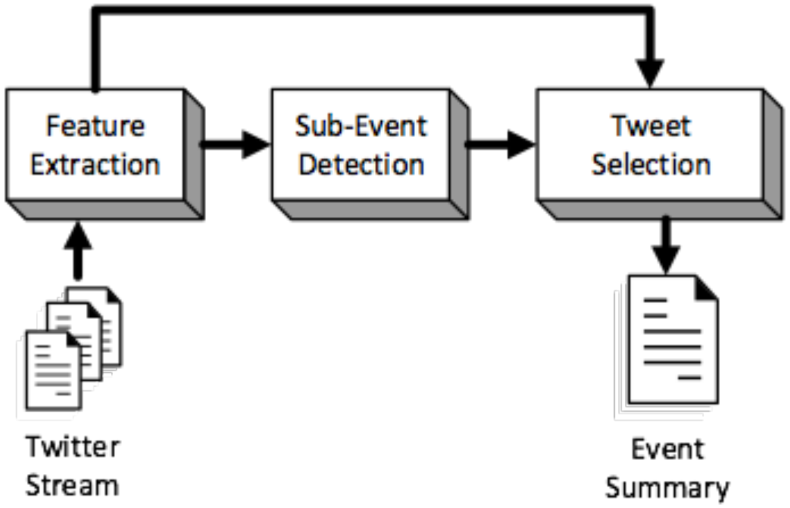
# System Architecture

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

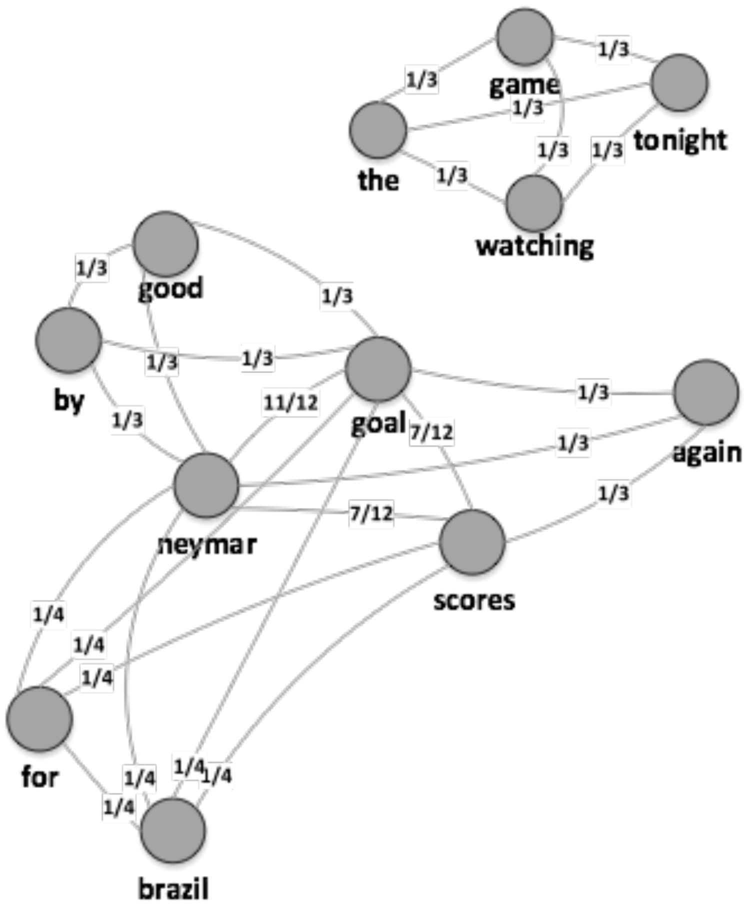
➢Represents all the input tweets
[Rousseau and Vazirgiannis, CIKM 2013]

➢node ←--→ unique term

➢edge ←--→ #co-occurrences within a tweet (normalized)

Example Graph:

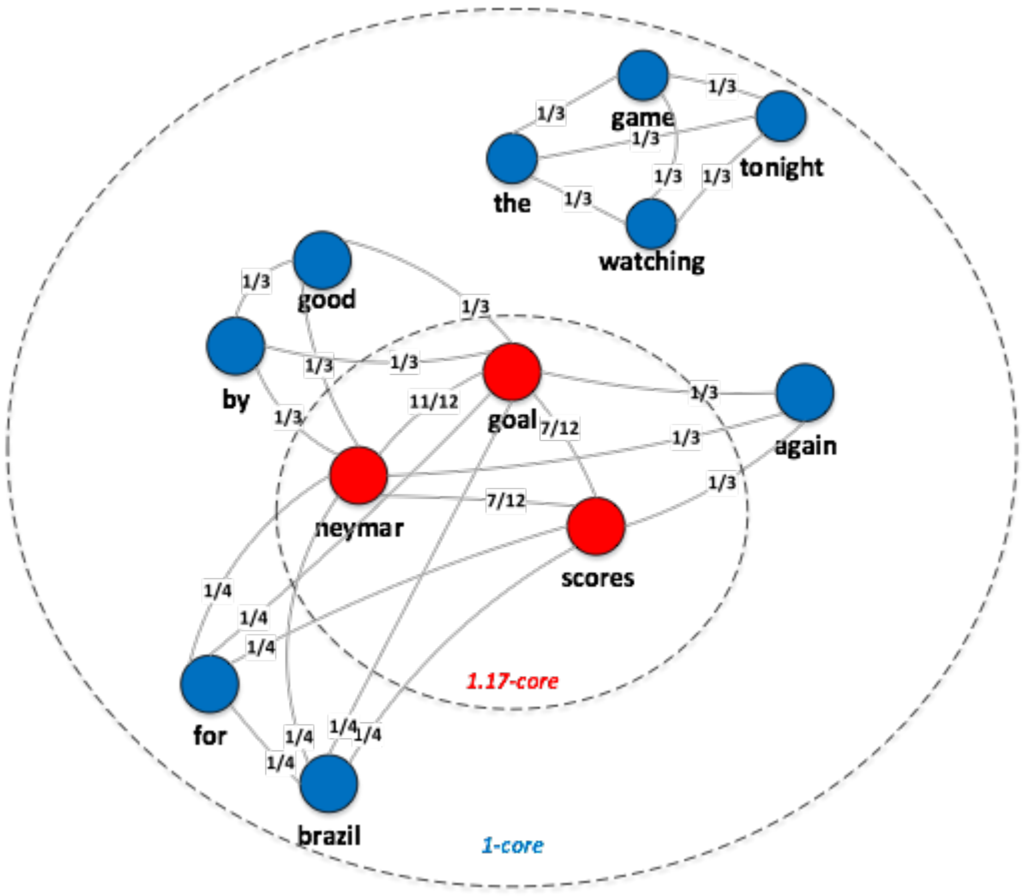
1) Good goal by Neymar
2) Goal! Neymar scores for brazil
3) Goal!! Neymar scores again
4) Watching the game tonight



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

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

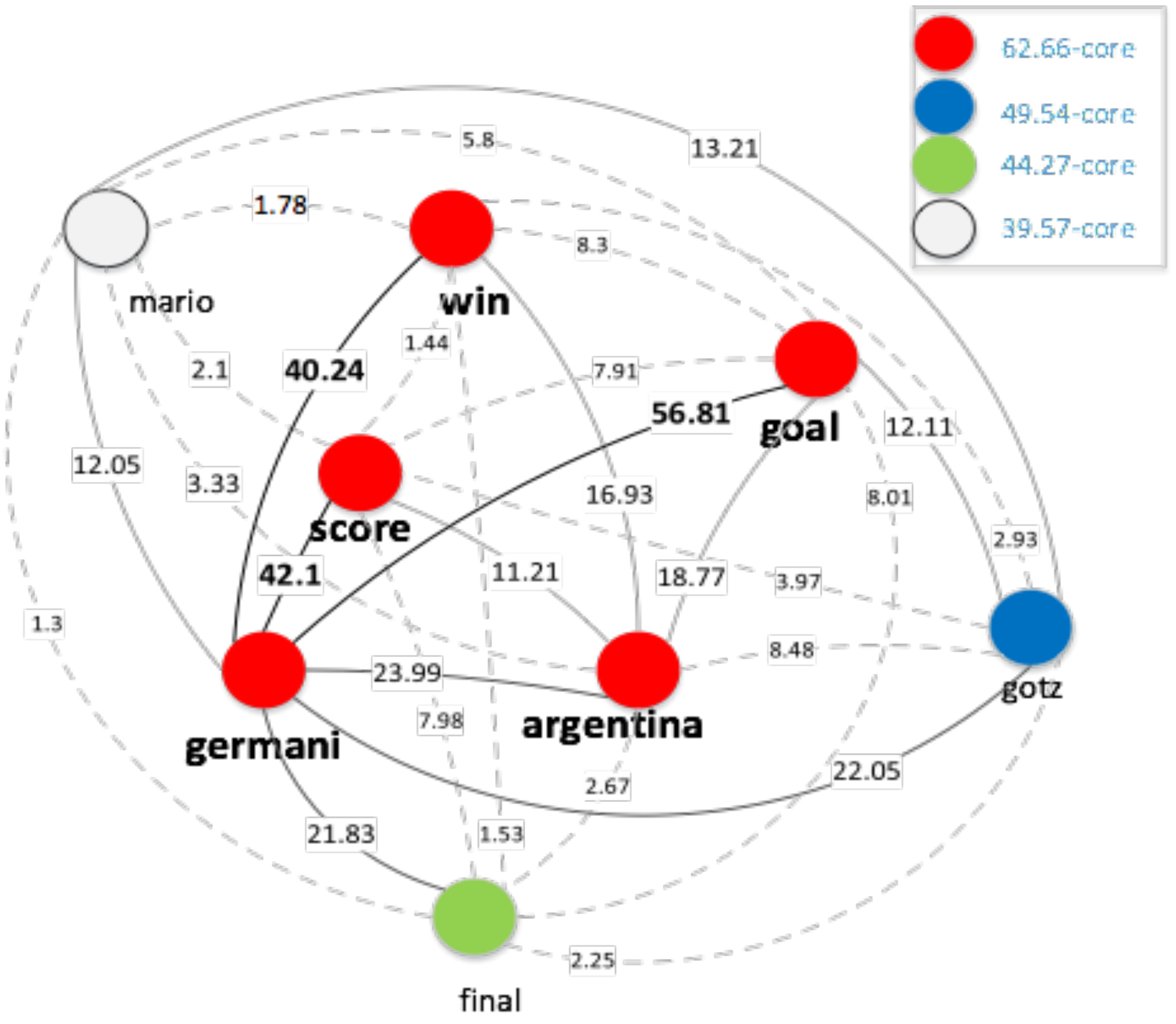
|  |  |
|---|---|
| $c_i^t$ | Core number of term $i$ at time slot $t$ |
| $d$ | Number of terms selected |
| $\theta$ | Decision Threshold |
| $p$ | Number of previous time slots |

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

$$\begin{tikzcd}[column sep=tiny] & \text{Sub-Event Detection} \arrow[dl] \arrow[dr] & \\ \text{Tweet Rate} & & \text{Term Weights} \end{tikzcd}$$

$$\begin{tikzcd}[column sep=tiny] & \text{Sub-Event Summarization} \arrow[dl] \arrow[dr] & \\ \text{Frequency of terms} & & \text{Core number of terms} \end{tikzcd}$$

Baselines-Approaches

(Detection -Term Weight)

➢ Rate–Freq: the common baseline

➢ Rate–Core

➢ ***Weight–Core***: Our approach

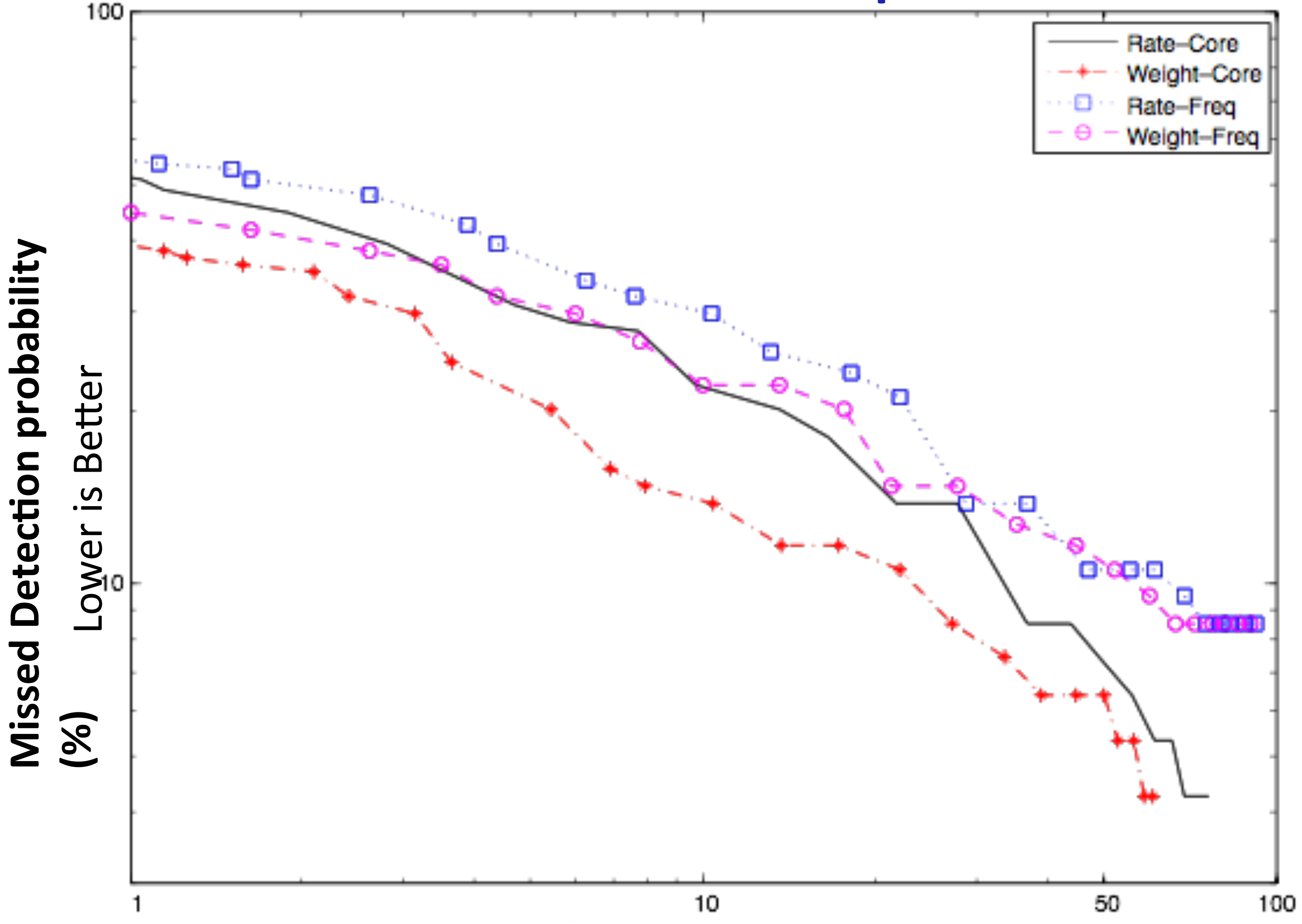
➢ Weight–Freq

## 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



**False Alarm probability (%)**

Lower is Better

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

⇒ 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].

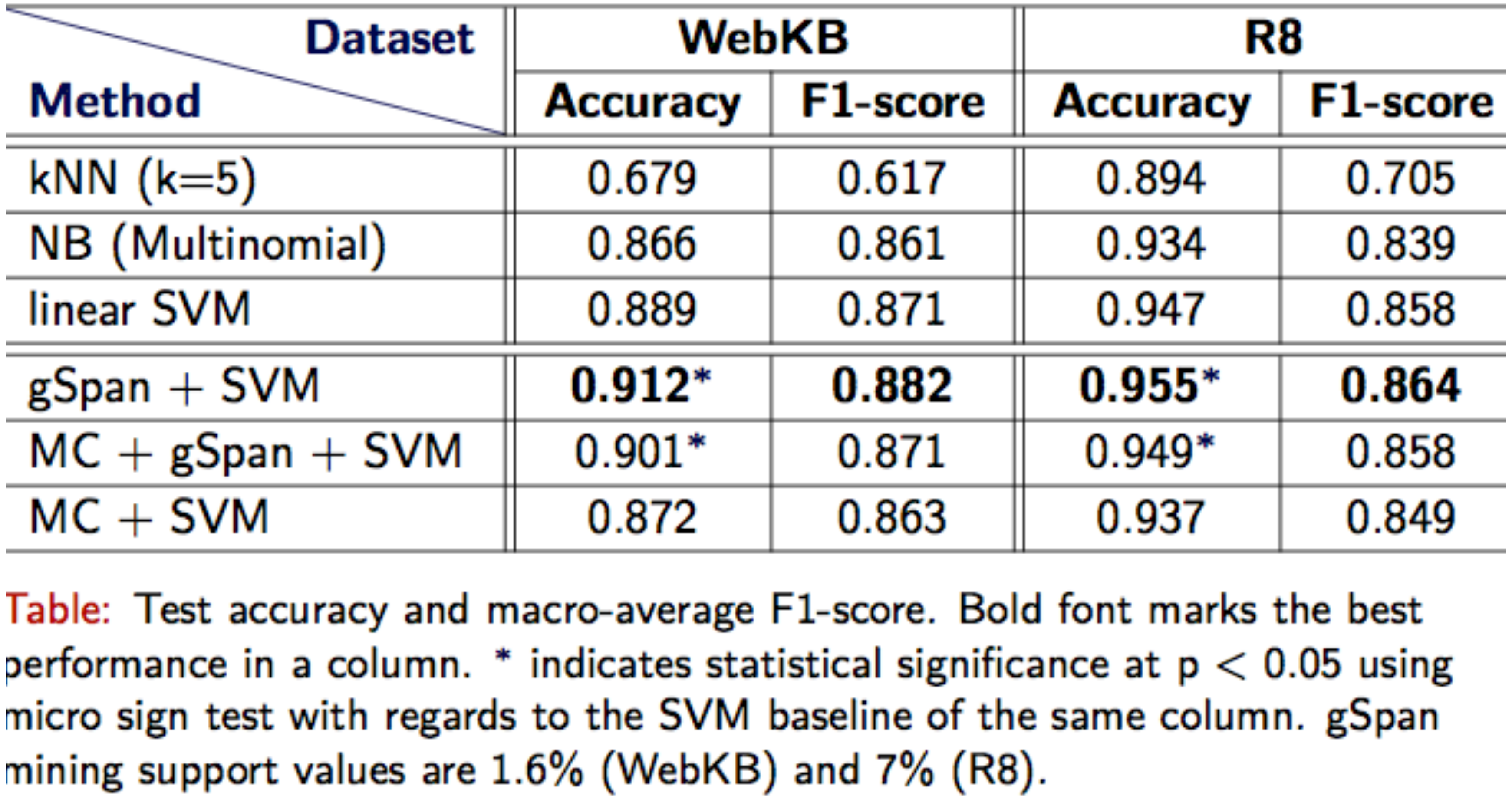
# Models

- 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

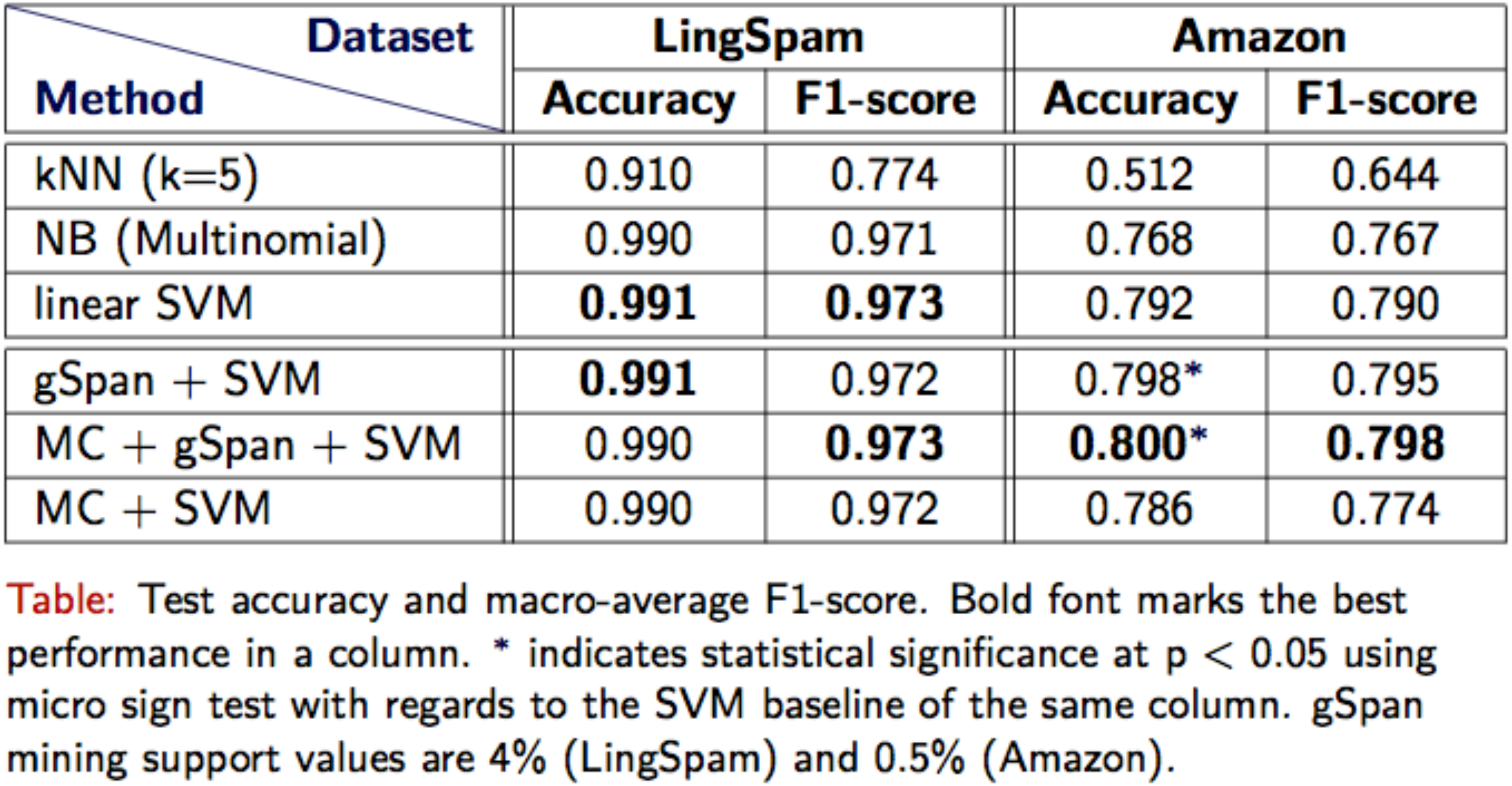
# Effectiveness results (1/2)



| Dataset / Method | 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).

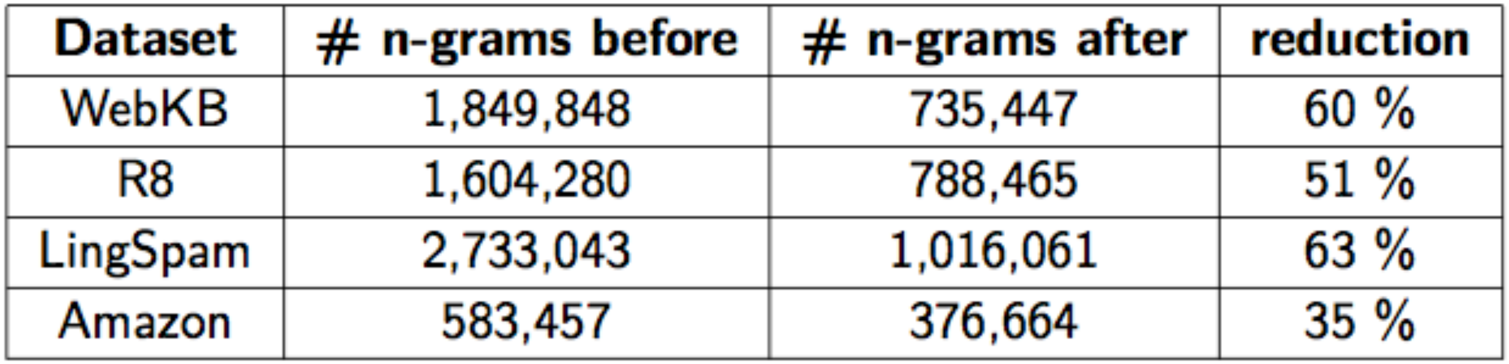
# Effectiveness results (2/2)



| 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

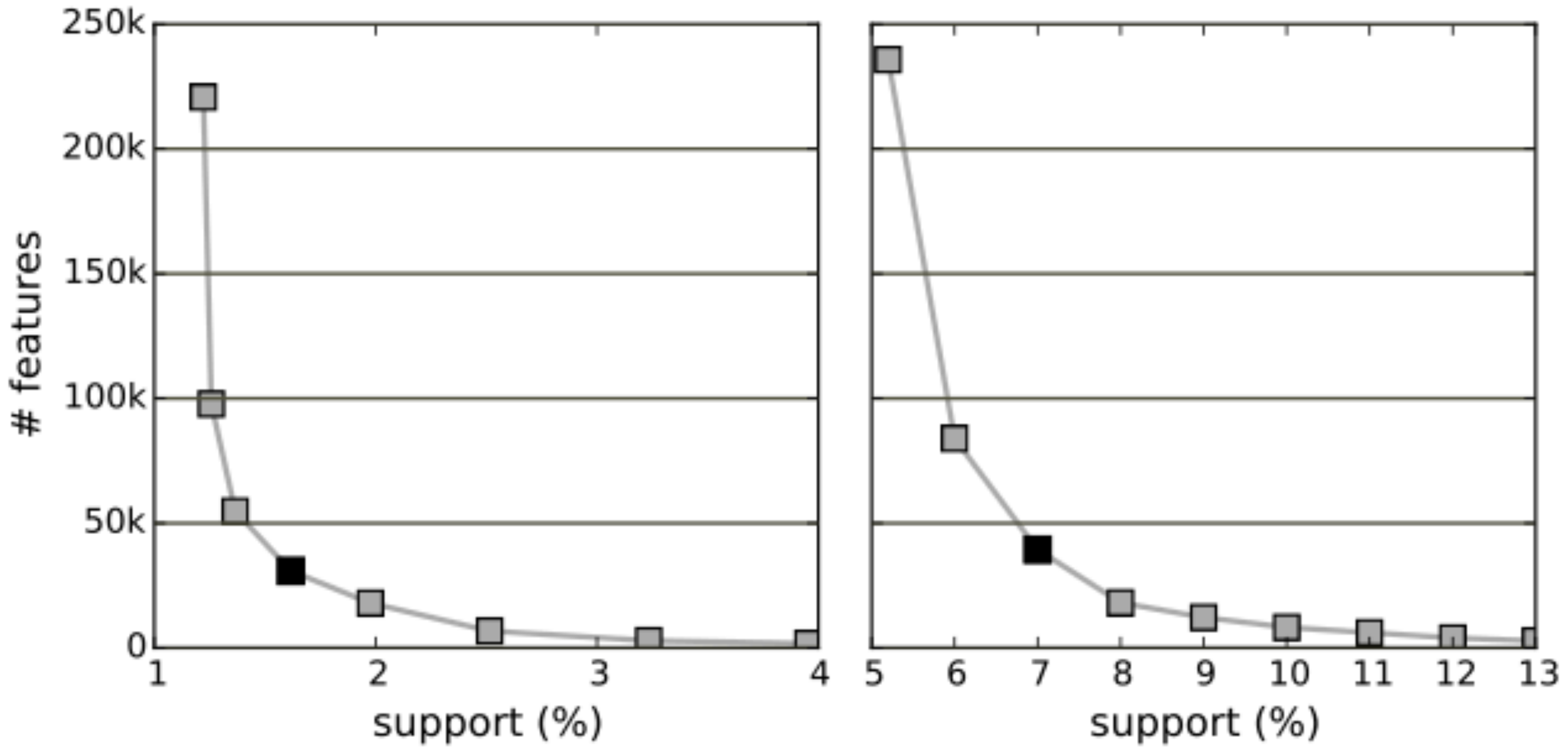


Figure: Number of subgraph features per support (%) on WebKB (left) and R8 (right) datasets. In black, the selected support chosen via the elbow method.

# Distribution of mined n-grams

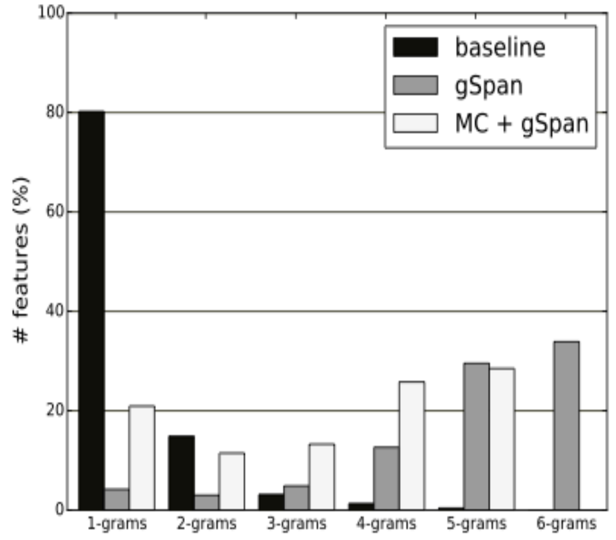


Figure: Distribution of n-grams (standard and long-distance ones) among all the features on WebKB dataset.

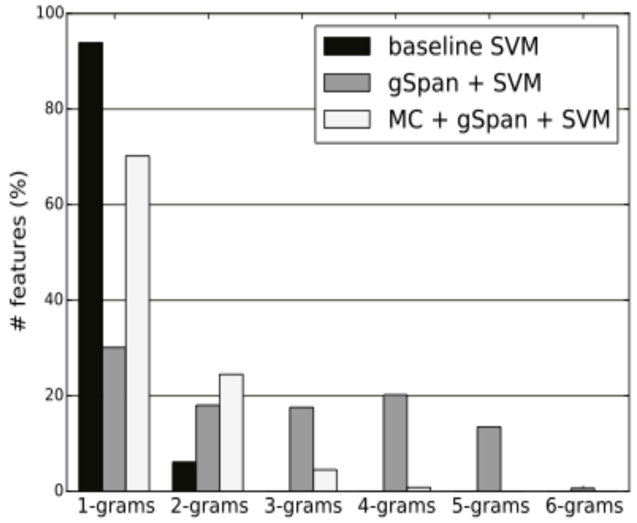


Figure: Distribution of n-grams (standard and long-distance ones) among the top 5% most discriminative features for SVM on WebKB dataset.

# 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.

# 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

## Sentence based Kernels for short text categorization **[EMNLP 2015]**

*Short text categorization*

Assume the sentences:

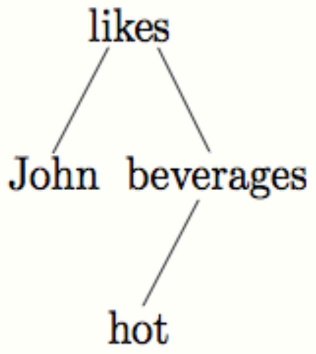
- 'John likes hot beverages'
- 'John loves warm drinks'

Their semantic similarity is obvious …

Challenge: devise a similarity function that finds adequate similarity in these cases

# Dependency trees

- Use of dependency tree structure to capture better bigrams

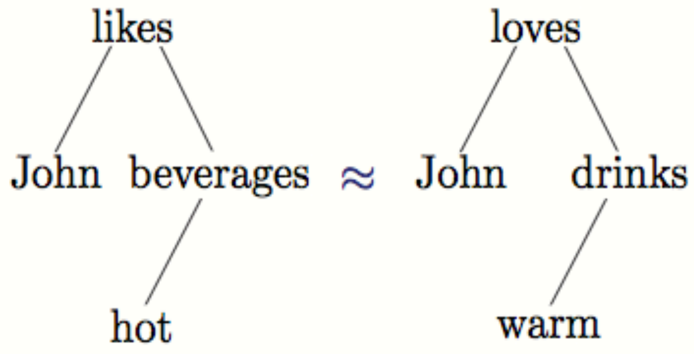


$K_b(\text{'John likes hot beverages'}, \text{'John likes warm beverages'}) = 3 + 2$

- Still problematic when inner nodes are different!

$K_b(\text{'John likes hot beverages'}, \text{'John loves warm drinks'}) = 1$

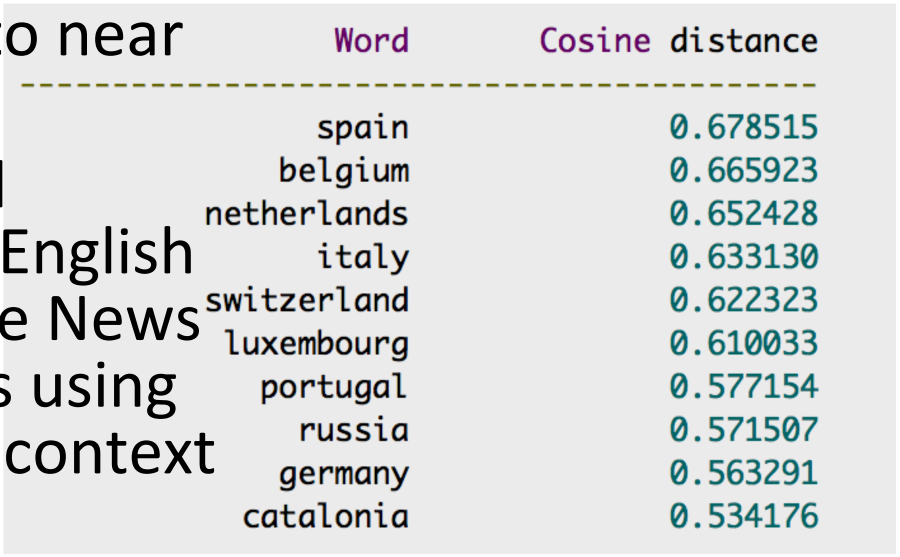
but



⇒ Sparsity issue for classification of *short documents* or *when little training data is available*

# Context

- Word embeddings
  - Embed words in a Low-dimension Euclidean vector space, suited for word similarity
  - Similar words are projects to near by places
  - word2vec: 300 dimensional representation of 3 million English words trained over a Google News dataset of 100 billion words using the Skip-gram model and a context size of 5
- Google word2vec [1]: query 'France', *distance* will display the most similar words and their distances to 'france'



```
                                Word       Cosine distance
------------------------------------------------------------
                               spain              0.678515
                             belgium              0.665923
                         netherlands              0.652428
                               italy              0.633130
                         switzerland              0.622323
                          luxembourg              0.610033
                            portugal              0.577154
                              russia              0.571507
                             germany              0.563291
                           catalonia              0.534176
```

[1] https://code.google.com/p/word2vec/

# Mathematical framework (i)

- **Word Kernel (WK)**

We define a kernel between two words as a polynomial kernel - cosine similarity in the word embedding space, where α is a scaling factor:

$$\mathrm{WK}(w_1, w_2) = \left[\frac{1}{2}\left(1 + \frac{\langle \mathbf{w}_1, \mathbf{w}_2 \rangle}{\|\mathbf{w}_1\|\|\mathbf{w}_2\|}\right)\right]^{\alpha}$$

- **Phrase Kernel (PhK)**

We considered two types of phrases:
- *co-occurrence phrases*: contiguous sequences of words in the original sentence
- *syntactic phrases*: downward paths in the dependency tree representation

**- Product Kernel (PK)**

$w_{ij}$ *i-th* word in phrase $p_j$ of length $i$

$$\mathrm{PK}(p_1, p_2) = \prod_{i=1}^{l} \mathrm{WK}(w_i^1, w_i^2)$$

- **Composition Kernel (CK)**

$$\mathrm{CK}(p_1, p_2) = \mathrm{WK}(\mathbf{p}_1, \mathbf{p}_2)$$

(in case of different sentence length) embedding **p***j* of the phrase *pj* obtained either by

- addition: $\mathbf{p}_j = (\sum_{i=1}^{l} \mathbf{w}_i^j)$
- or element wise multiplication $\mathbf{p}_j = (\bigodot_{i=1}^{l} \mathbf{w}_i^j)$ i.e.

i.e. 'Berlin' with 'capital of Germany'

# Mathematical framework II

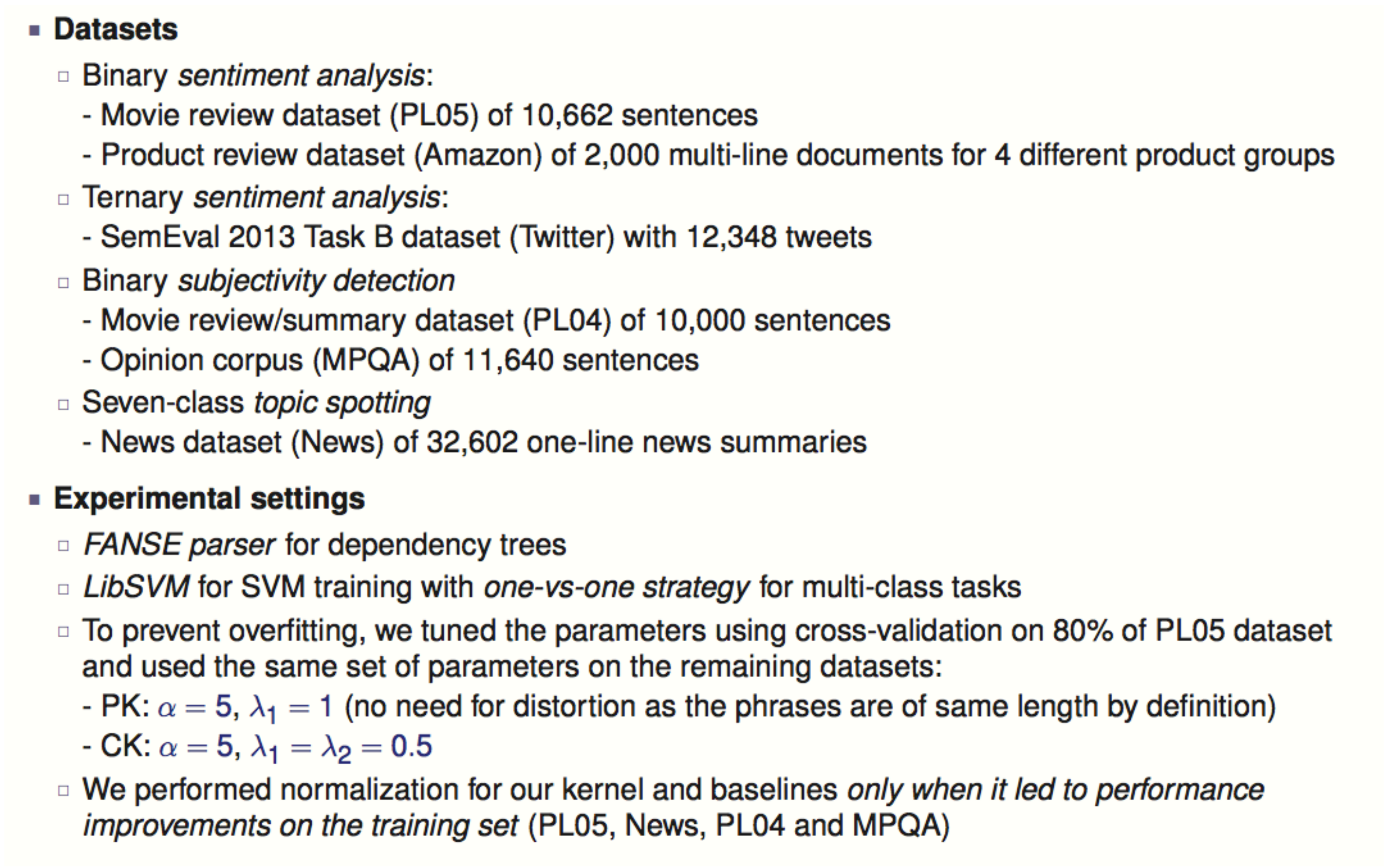
- **Sentence Kernel (SK)**
  - Defined through convolution as the sum of all local phrasal similarities, i. e. kernel values between phrases contained in the sentences:

$$\mathrm{SK}(s_1, s_2) = \sum_{\substack{p_1 \in \phi(s_1),\\ p_2 \in \phi(s_2)}} \lambda_1{}^{\epsilon}\, \lambda_2{}^{\eta}\, \mathrm{PhK}(p_1, p_2)$$

  where
  - $\phi(s_k)$: set of phrases in sentence $s_k$ (co-occurrence or syntactic phrases)
  - $\lambda_1$: decaying factor penalizing longer phrases
  - $\epsilon = \max\{|p_1|, |p_2|\}$: the maximum length of the two phrases
  - $\lambda_2$: distortion parameter
  - $\eta = ||p_1| - |p_2||$: length difference between the two phrases
  - PhK: phrase kernel (either PK, $\mathrm{CK}^{+}$ or $\mathrm{CK}^{\odot}$)

- **Document Kernel**
  - Sum of all sentence kernel values between two documents
  - We may normalize each kernel value $K_{ij}$ between documents *i* and *j* by $\sqrt{K_{ii}K_{jj}}$

# Experiments (i)

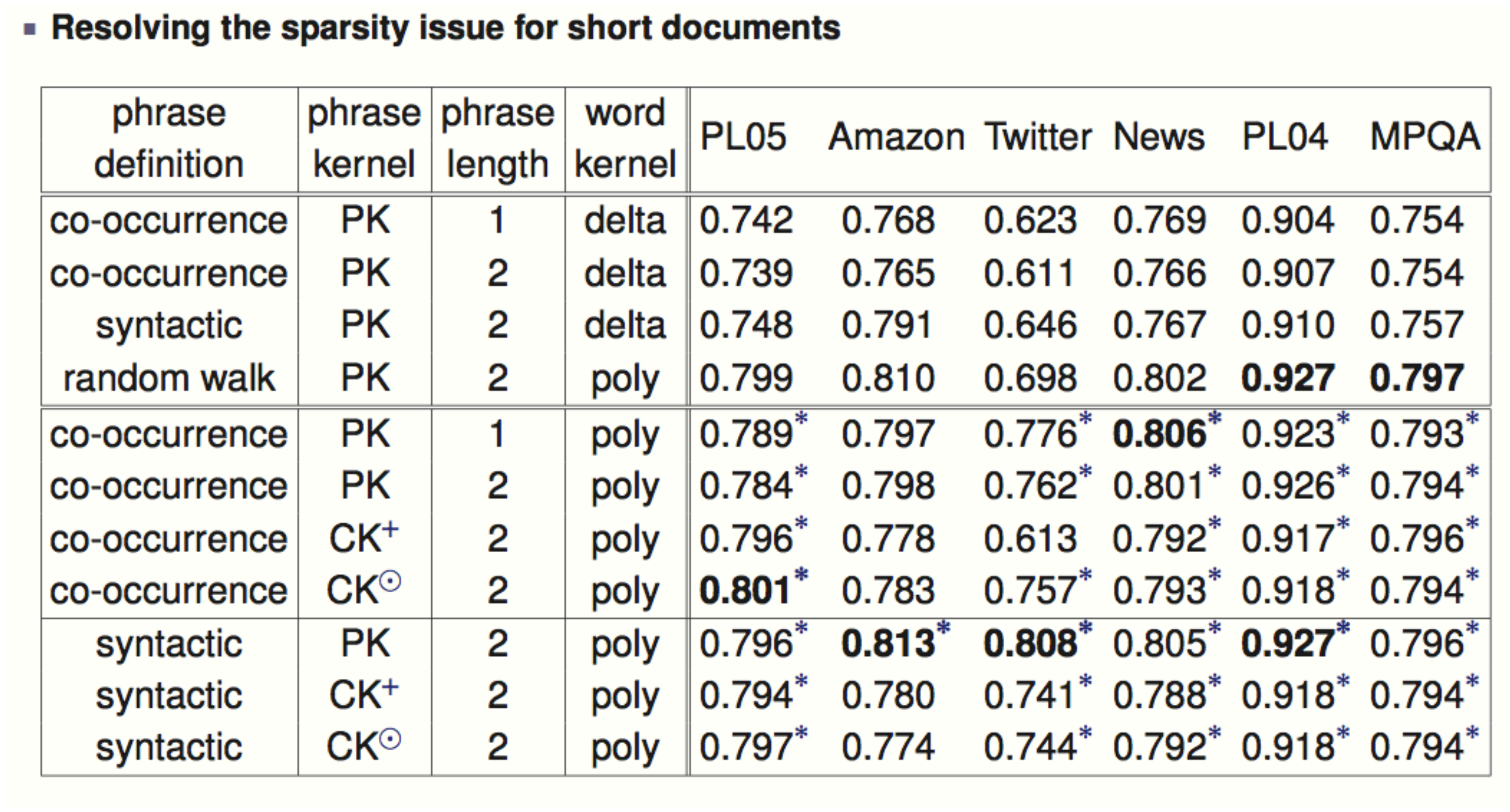


- **Datasets**
  - Binary *sentiment analysis*:
    - Movie review dataset (PL05) of 10,662 sentences
    - Product review dataset (Amazon) of 2,000 multi-line documents for 4 different product groups
  - Ternary *sentiment analysis*:
    - SemEval 2013 Task B dataset (Twitter) with 12,348 tweets
  - Binary *subjectivity detection*
    - Movie review/summary dataset (PL04) of 10,000 sentences
    - Opinion corpus (MPQA) of 11,640 sentences
  - Seven-class *topic spotting*
    - News dataset (News) of 32,602 one-line news summaries
- **Experimental settings**
  - *FANSE parser* for dependency trees
  - *LibSVM* for SVM training with *one-vs-one strategy* for multi-class tasks
  - To prevent overfitting, we tuned the parameters using cross-validation on 80% of PL05 dataset and used the same set of parameters on the remaining datasets:
    - PK: $\alpha = 5$, $\lambda_1 = 1$ (no need for distortion as the phrases are of same length by definition)
    - CK: $\alpha = 5$, $\lambda_1 = \lambda_2 = 0.5$
  - We performed normalization for our kernel and baselines *only when it led to performance improvements on the training set* (PL05, News, PL04 and MPQA)

# Experiments (ii)

- **Evaluation**
  - We report *accuracy* on:
    - PL05: remaining 20% test set
    - Twitter: standard test split (25%)
    - News: standard test split (50%)
    - Amazon, PL04, MPQA: 5-fold CV
  - Significance test: *micro sign test* ($p < 0.01$) over the bigram baseline with same phrase definition
- **Models**
  - Delta word kernel
    - standard unigram and bigram baseline approaches
  - Dependency Tree Kernel (DTK)
    - $n$-grams from downward paths in dependency trees
  - Vector Tree Kernel (VTK)
    - $\phi(\cdot)$ enumerates all random walks in the dependency tree
  - Our convolutional sentence kernels based on phrase kernels and polynomial word kernels

# Results (i)

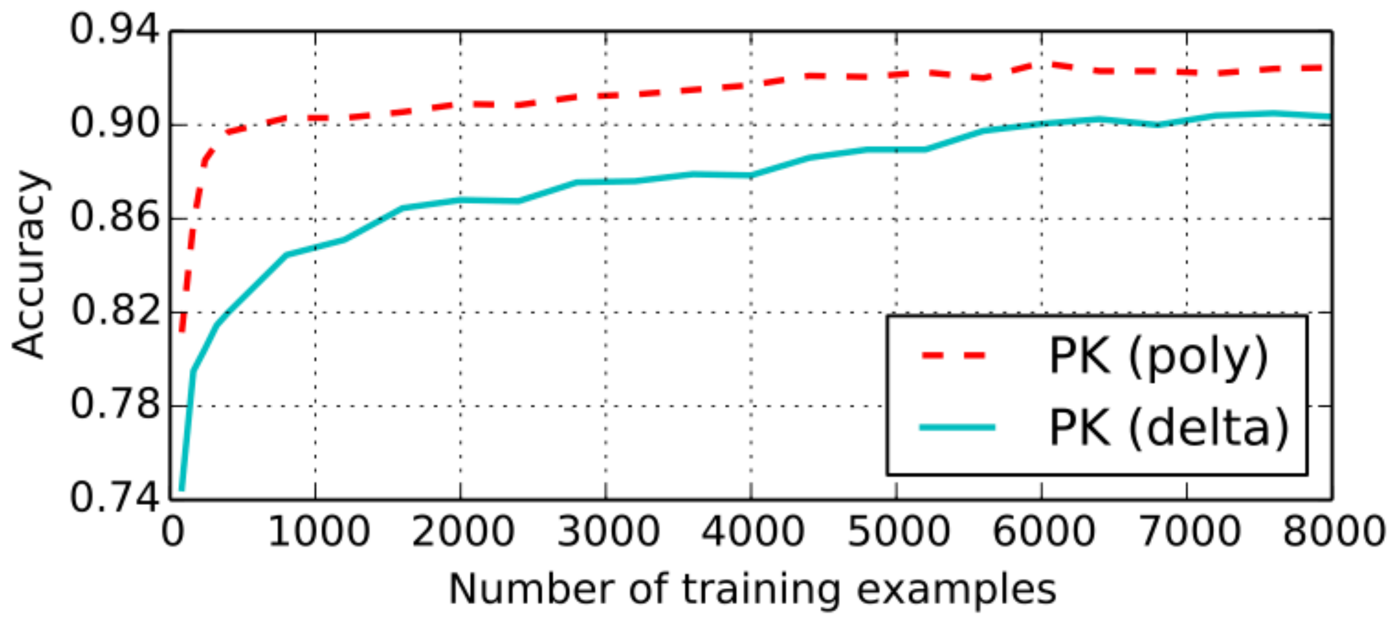


- **Resolving the sparsity issue for short documents**

| phrase definition | phrase kernel | phrase length | word kernel | PL05 | Amazon | Twitter | News | PL04 | MPQA |
|---|---|---|---|---|---|---|---|---|---|
| co-occurrence | PK | 1 | delta | 0.742 | 0.768 | 0.623 | 0.769 | 0.904 | 0.754 |
| co-occurrence | PK | 2 | delta | 0.739 | 0.765 | 0.611 | 0.766 | 0.907 | 0.754 |
| syntactic | PK | 2 | delta | 0.748 | 0.791 | 0.646 | 0.767 | 0.910 | 0.757 |
| random walk | PK | 2 | poly | 0.799 | 0.810 | 0.698 | 0.802 | **0.927** | **0.797** |
| co-occurrence | PK | 1 | poly | 0.789* | 0.797 | 0.776* | **0.806*** | 0.923* | 0.793* |
| co-occurrence | PK | 2 | poly | 0.784* | 0.798 | 0.762* | 0.801* | 0.926* | 0.794* |
| co-occurrence | $CK^+$ | 2 | poly | 0.796* | 0.778 | 0.613 | 0.792* | 0.917* | 0.796* |
| co-occurrence | $CK^{\odot}$ | 2 | poly | **0.801*** | 0.783 | 0.757* | 0.793* | 0.918* | 0.794* |
| syntactic | PK | 2 | poly | 0.796* | **0.813*** | **0.808*** | 0.805* | **0.927*** | 0.796* |
| syntactic | $CK^+$ | 2 | poly | 0.794* | 0.780 | 0.741* | 0.788* | 0.918* | 0.794* |
| syntactic | $CK^{\odot}$ | 2 | poly | 0.797* | 0.774 | 0.744* | 0.792* | 0.918* | 0.794* |

## Results (ii)

- **Resolving the sparsity issue when little training data is available**
  - Learning done on 1% to 100% of the training set (80% of the dataset) on PL04
    - ⇒ our kernel starts to plateau earlier
    - ⇒ our kernel reaches the maximum baseline accuracy with only 1,500 training examples



# Related efforts

**Kernels**

- *semantic kernels* for text categorization, capitalizing on WordNet: continuous word kernels based on the inverse of the path lengths in the tree rather than the common delta word kernel used so far, i.e. exact matching between unigrams [Siolas, d'Alché Buc. 2000]
- extensions to tree similarity measures [Bloehdorn et al. 2006]
- GVSM kernels [Mavroeidis, Vazirgiannis et.al 2005]
- *tree kernels* to compare trees based on their topology (e.g., shared subtrees) rather than the similarity between their nodes [Collins, Duffy 2001]
- Dependency Tree Kernels (DTK) to capture syntactic similarities [Culotta and Sorensen 2004]
- additional semantic meta-features based on word embeddings to enhance classification for short text similarity [Song and Roth (2015)[Kenter, Rijke (2015)]

**Word Embeddings**

- Wordnet [Miller1995],
- SENNA[Collobert et al., 2011]
- Word2vec [Mikolov et al.2013]

## Conclusions

- sparsity issue  classifying short documents or when little training data is available
- we proposed a novel convolutional sentence kernel based on word embeddings that tackles sparsity issue.
- achieved significant improvements over the base-lines across all datasets when taking into account the additional information from the latent word similarity (word embeddings) and the syntactic structure (dependency tree).

# 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 …
- Future work
- designing new kernels for syntactic parse trees – similarity between non-terminal nodes

## Novel models needed – Thought vectors

- thought vectors": Google is working on a new type of algorithm designed to encode thoughts as sequences of numbers
- Geoff Hinton – Google: "If you take the vector for Paris and subtract the vector for France and add Italy, you get Rome," he said. "It's quite remarkable." @Guardian

https://wtvox.com/robotics/google-is-working-on-a-new-algorithm-thought-vectors

# Acknowledgements

**Collaborators**

F. Rousseau(LIX), P. Meladianos (AUEB), Y. Nikoletzos (AUEB), E. Kiagias (AUEB), J. Kim (Oxford)

**Funding**

Google, Airbus, DIGITEO, AXA (Chair)

# 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

THANK YOU!

Questions?

http://www.lix.polytechnique.fr/dascim/

**We are hiring!!**

ML for graph and text mining

postdoc and PhD

Contact me: mvazirg@lix.polytechnique.fr