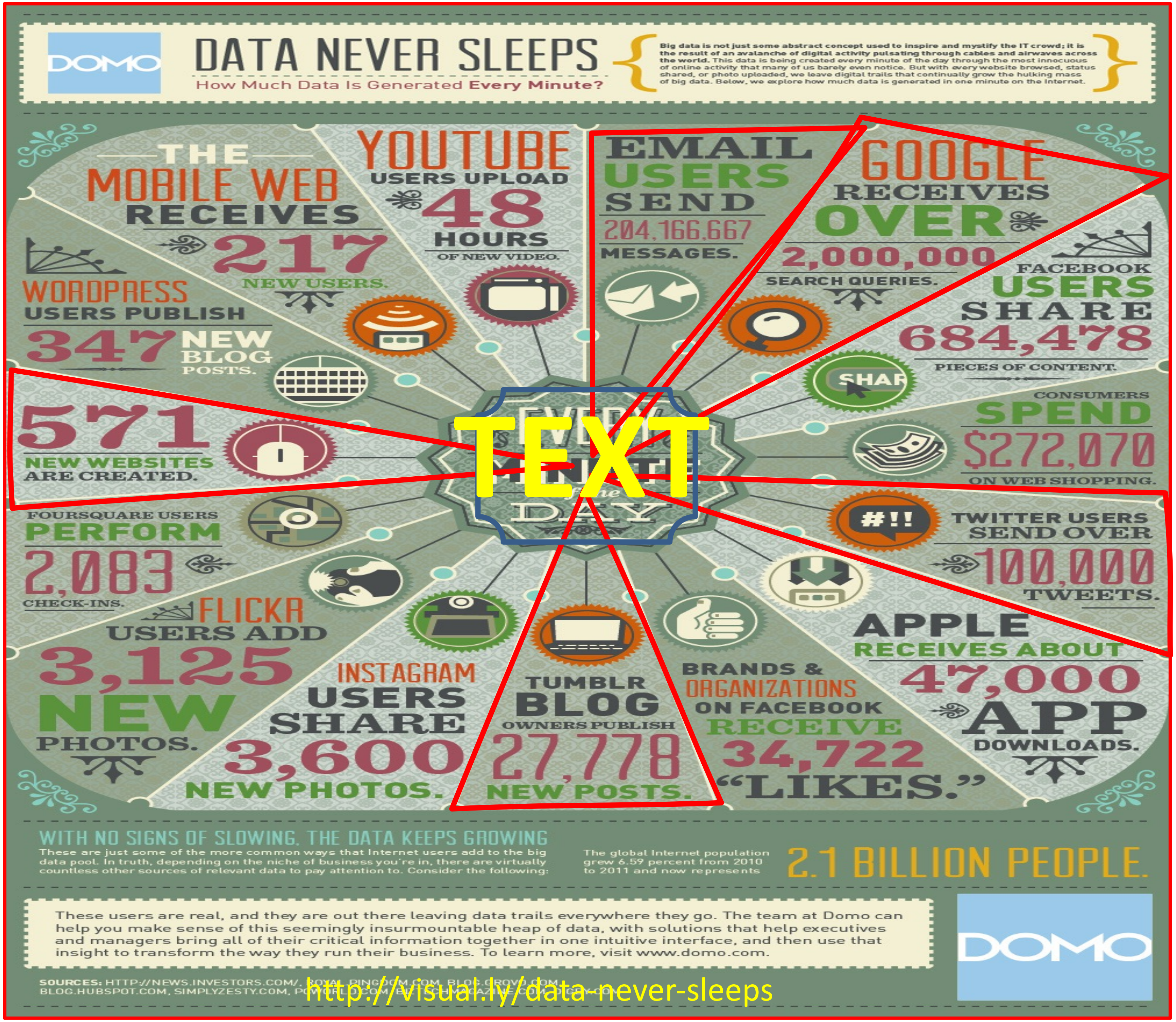
NLP – Information retrieval, attention for machine translation

Prof. Michalis Vazirgiannis
LIX @ Ecole Polytechnique
October 2023
https://scholar.google.gr/citations?user=aWGJYcMAAAAJ&hl=el

Outline

- **TF/IDF - Text Retrieval**
- Graph Based IR and keyword extraction
- Attention in deep learning for NLP
- RNN for machine translation



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

Boolean Vector Model

- Boolean model
  - Text 1: ““This is the text mining course lecture”
  - Text 2: “This is an introductory text course”

	This	is	the	text	mining	course	Lecture	an	introductory
Text 1:	1	1	1	1	1	1	1	0	0
Text 2:	1	1		1	0	1	0	1	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	the	text	mining	course	Lecture	an	introductory
Text 1:	1/7	1/7	1/7	1/7	1/7	1/7	1/7	0	0
Text 2:	1/6	1/6	0	1/6	1/6	1/6	0	1/6	1/6

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.

6

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 $\in \{0, 1\}^{|V|}$.

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

[Introduction to Information Retrieval, C. Manning, P. Raghavan, H. Schütze, 978-0521865715, Cambridge Univ. Press]

7

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 $\in \mathbb{N}^{|V|}$.

[Introduction to Information Retrieval, C. Manning, P. Raghavan, H. Schütze, 978-0521865715, Cambridge Univ. Press]

8

Bag of words model

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

9

Term frequency: Log frequency weighting

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

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

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

$$tf(q, d) = \sum_{t \in q \cap d} (1 + \log tf_{t,d})$$

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

Frequency in document vs. frequency in collection

- 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 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 tf weight * idf.

$$w_{t,d} = (1 + logtf_{t,d}) * log\frac{N}{df_t}$$

- 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 $\in \mathbb{N}^{|V|}$.

[Introduction to Information Retrieval, C. Manning, P. Raghavan, H. Schütze, 978-0521865715, Cambridge Univ. Press]

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
. . .						

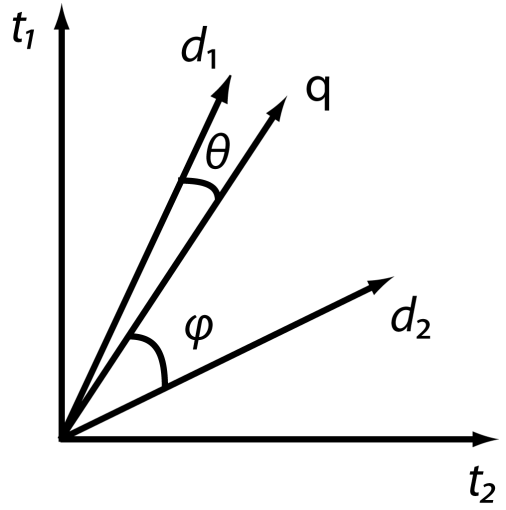
Each document is now represented as a real-valued vector of tf x idf weights $\in R^{|V|}$.

[Introduction to Information Retrieval, C. Manning, P. Raghavan, H. Schütze, 978-0521865715, Cambridge Univ. Press]

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 $(10 \times a, 10 \times b)$ 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 with tf-idf (and variations)

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

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

19

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

- query considered as a new document there fore represented as a vector.
- 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

Outline

- TF/IDF
- **Text Retrieval**
- Advanced tf weighting schemes

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 trade-off

- 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 relevant docs has 100% recall!
- The converse is also true (usually): It's easy to get high precision for very low recall.

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

27

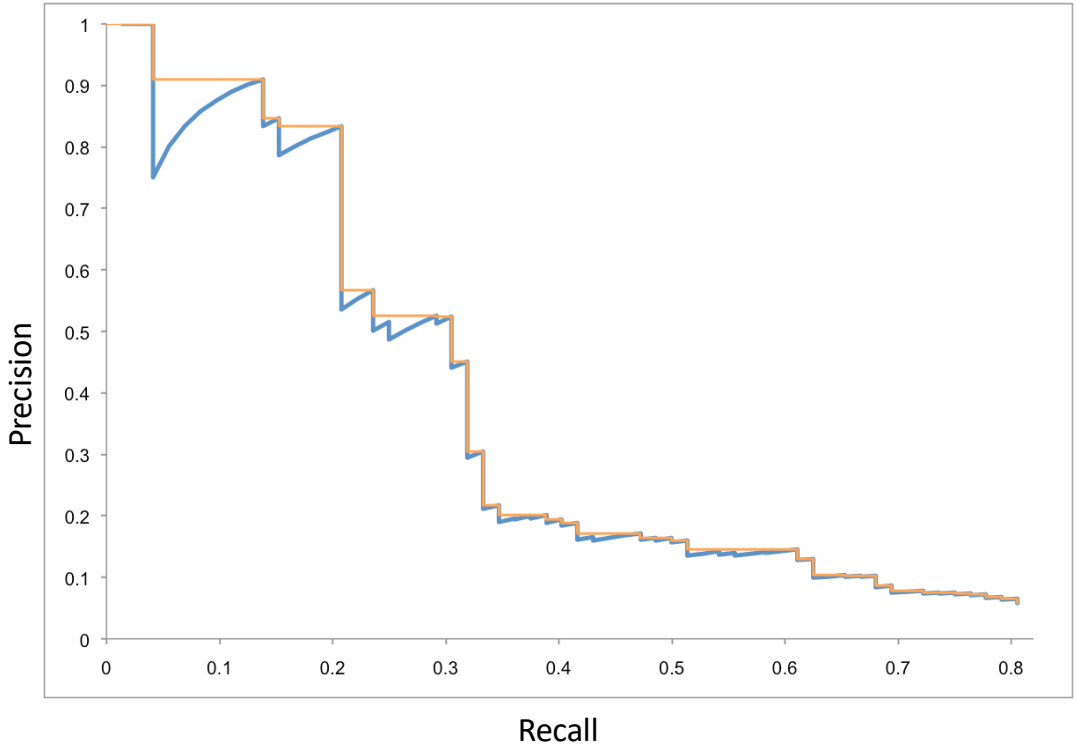
F: Example

	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]

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

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

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

Outline

- TF/IDF - Text Retrieval
- **Graph Based IR and keyword extraction**
- Attention in deep learning for NLP
- RNN for machine translation

Graph based IR and keyword extraction

- **Graph-of-words**
- Keyword extraction

Bag-of-Words (BoW) - Issues

Example text

information retrieval is the activity of obtaining

information resources relevant to an information need

from a collection of information resources

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

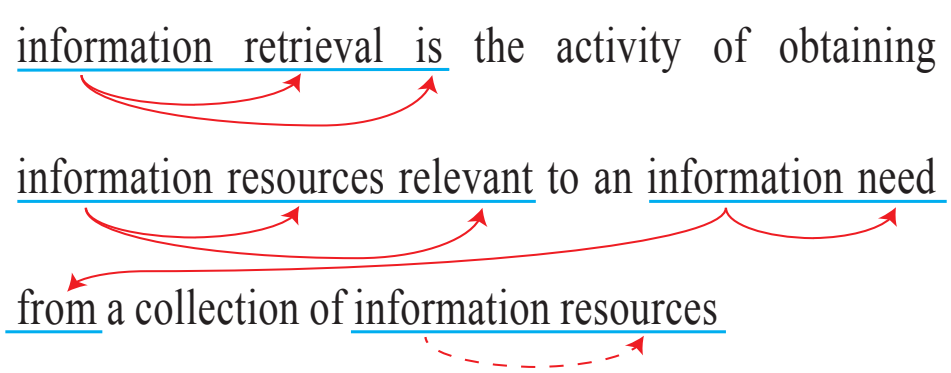
- Term independence assumption
- Term frequency weighting

Assumptions made by the BoW model

Graph-based Document Representation

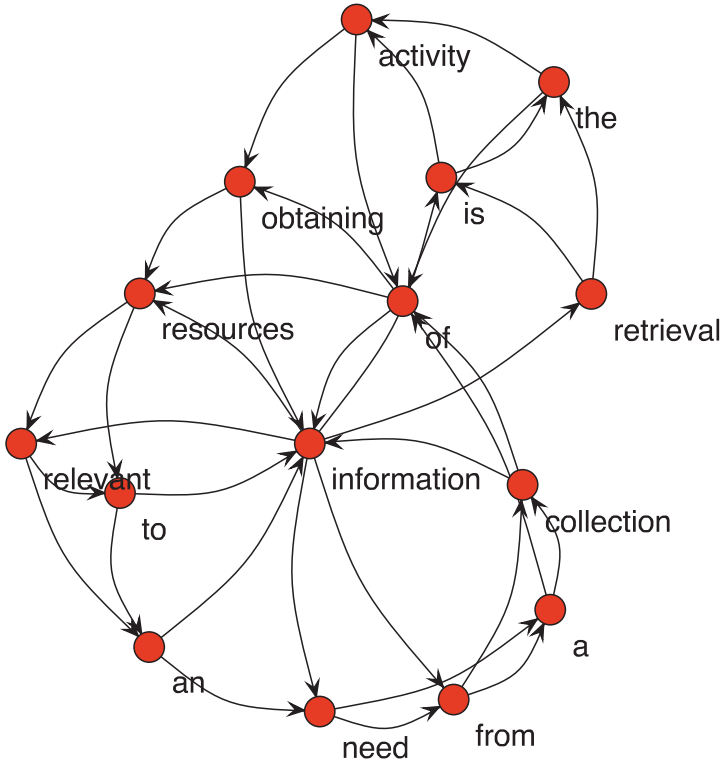
- Challenge the term independence and term frequency weighting assumptions taking into account word dependence, order and distance
- Employ a graph-based document representation capturing the above
- Graphs have been successfully used in IR to encompass relations and propose meaningful weights (e.g., PageRank)

Graph-based Document Representation - Example



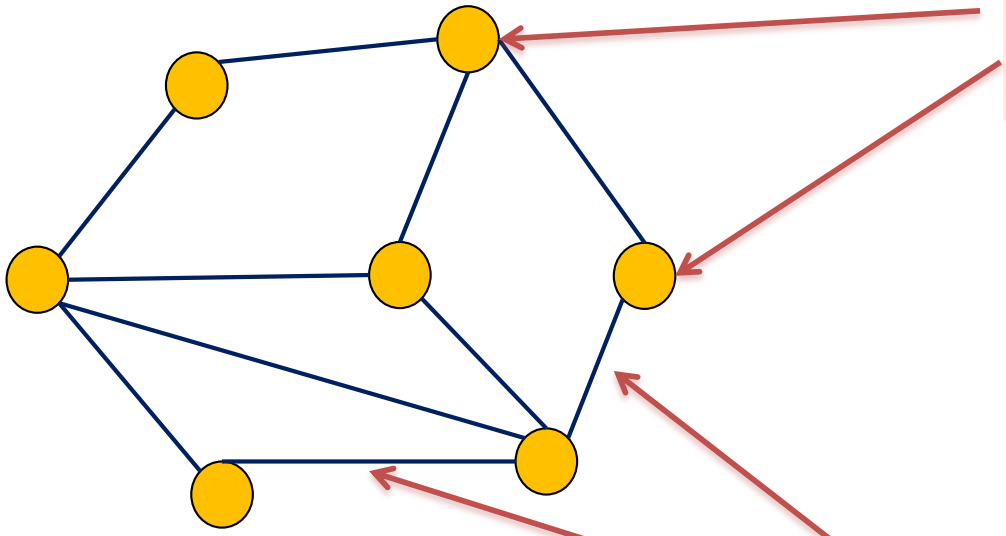
Idea: Replace term frequency with node centrality

Captures: frequency, order and distance



Graphs and Networks

Graphs: modeling dependencies



Nodes (or vertices)
(objects/entities)

Edges (or links)
(interconnections)

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

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

Centrality in Networks

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

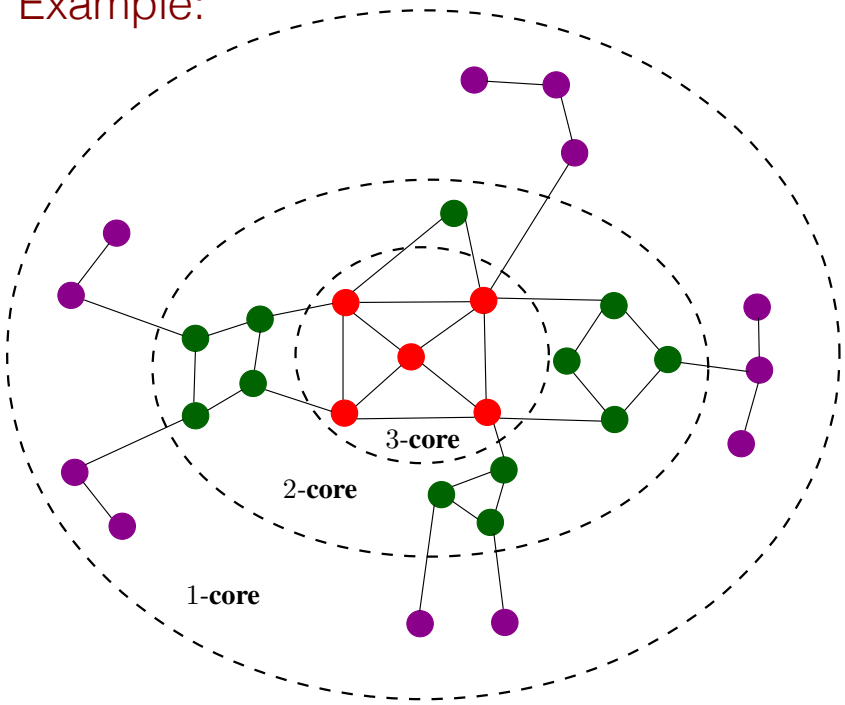
Core Decomposition

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

k-Core Decomposition

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

Example:



Core number $\mathbf{c_i = 1}$

Core number $\mathbf{c_i = 2}$

Core number $\mathbf{c_i = 3}$

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

$\mathbf{G_0 = G}$

$\mathbf{G_1}$ = 1-core of $\mathbf{G}$

$\mathbf{G_2}$ = 2-core of $\mathbf{G}$

$\mathbf{G_3}$ = 3-core of $\mathbf{G}$

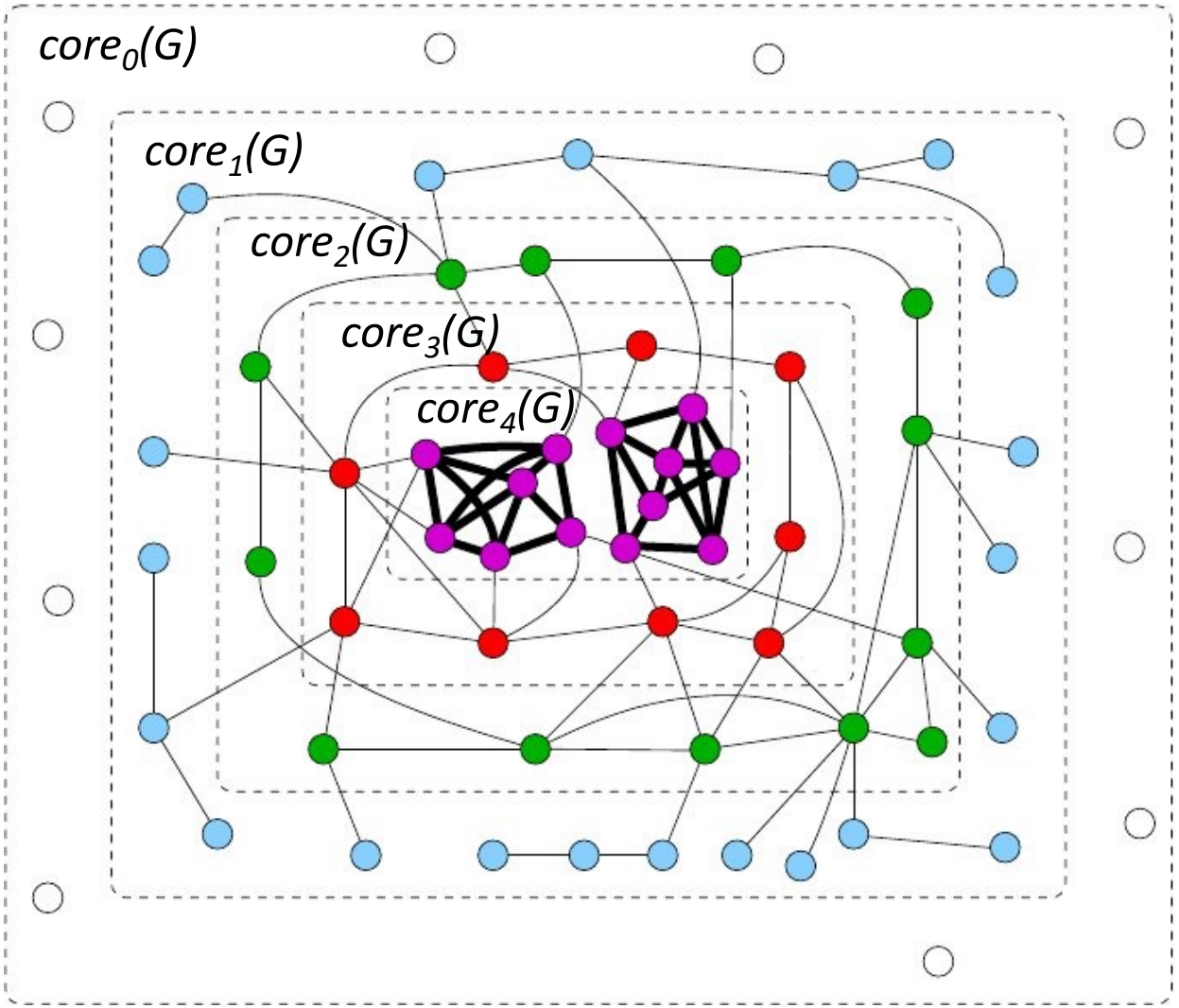
$\mathbf{G_0 \supseteq G_1 \supseteq G_2 \supseteq G_3}$

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

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

Also known as graph degeneracy

Another Example



Algorithm for k-Core Decomposition

Algorithm k-core(G, k)
Input: An undirected graph G and positive integer k
Output: k-core(G)
1. let F := G
2. while there is a node x in F such that $\deg_F(x)<k$
   delete node x from F
3. return F

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

[Batagelj and Zaversnik, '03]

Graph Semantics

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

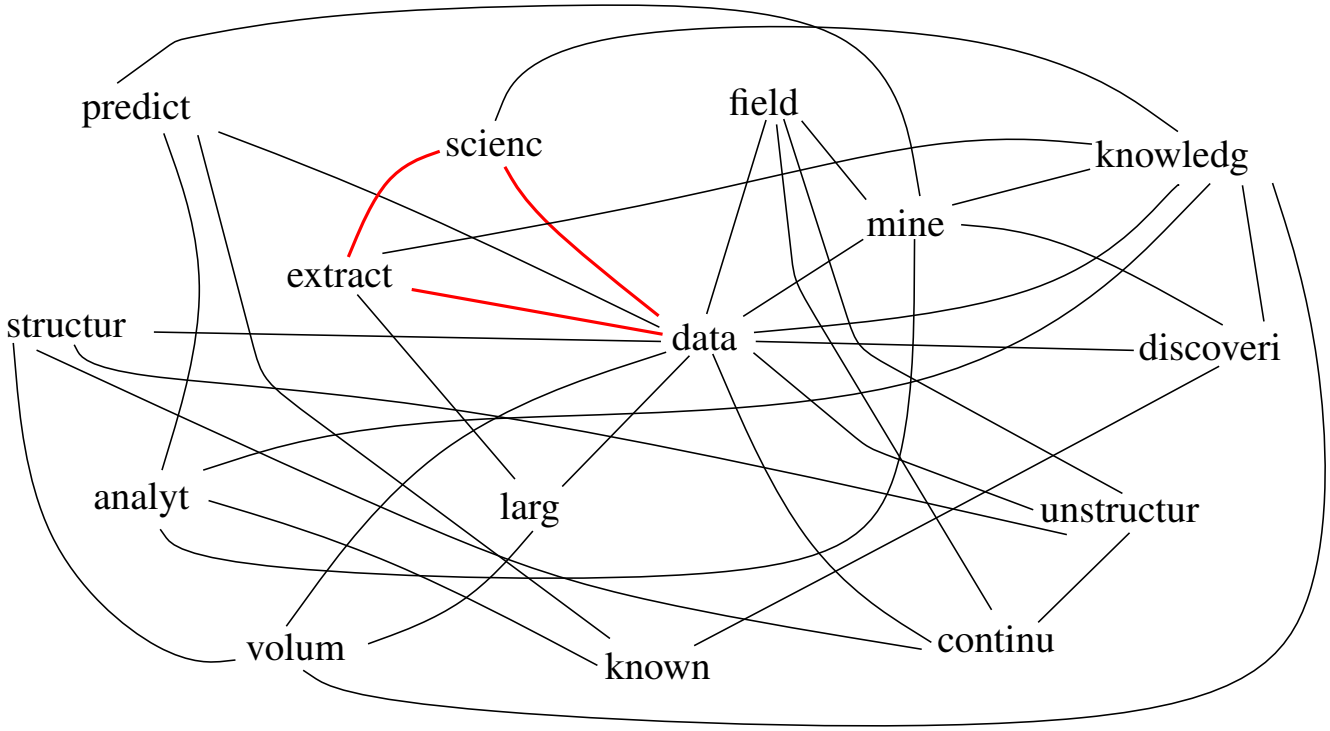
Graph-of-Words (GoW) Model

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

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

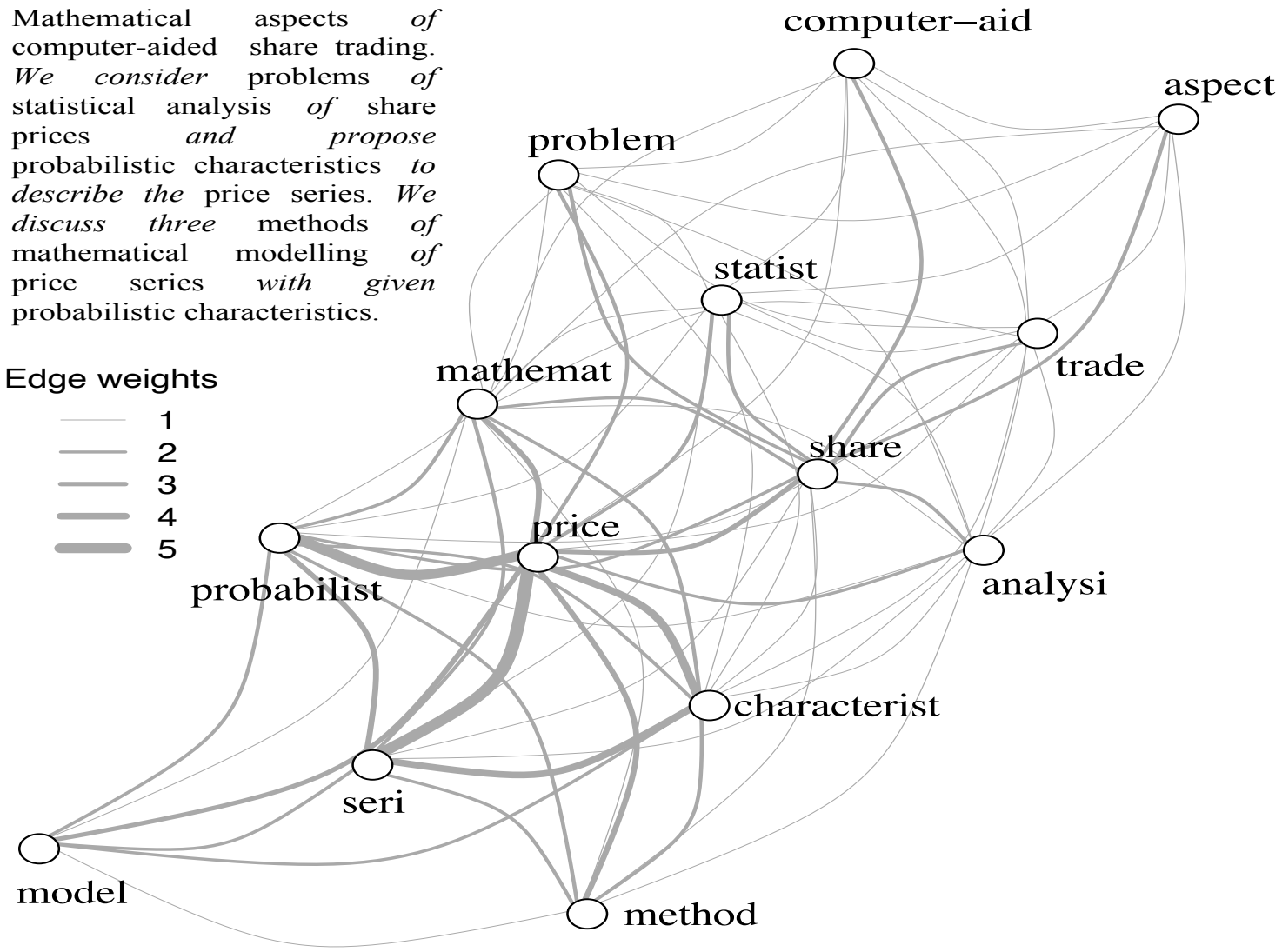
Example of Unweighted GoW

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



w = 3
unweighted, undirected graph

Example of Weighed Undirected GoW

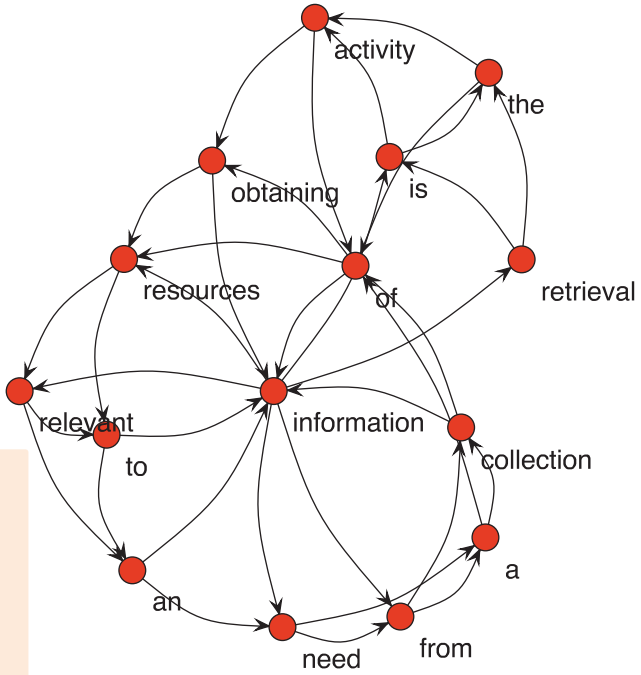


In-degree based TW

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

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

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



TF-IDF and BM25

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

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

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

- BM25 [Lv and Zhai, CIKM '11]

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

TW-IDF

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

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

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

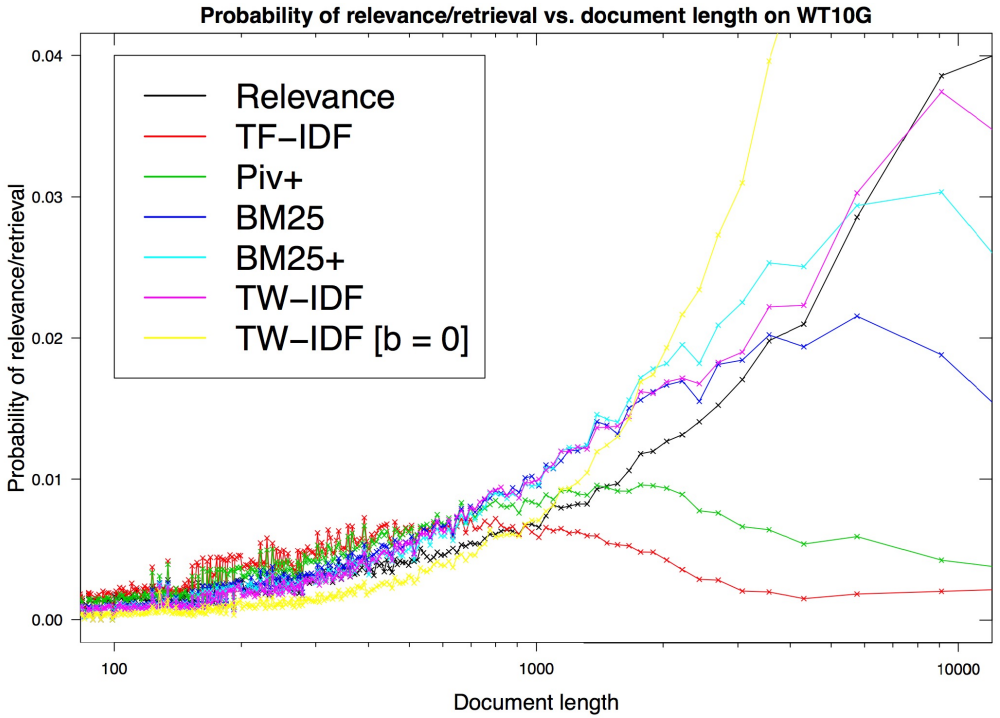
[Rousseau and Vazirgiannis, CIKM '13, best paper mention award]

Experimental Evaluation

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

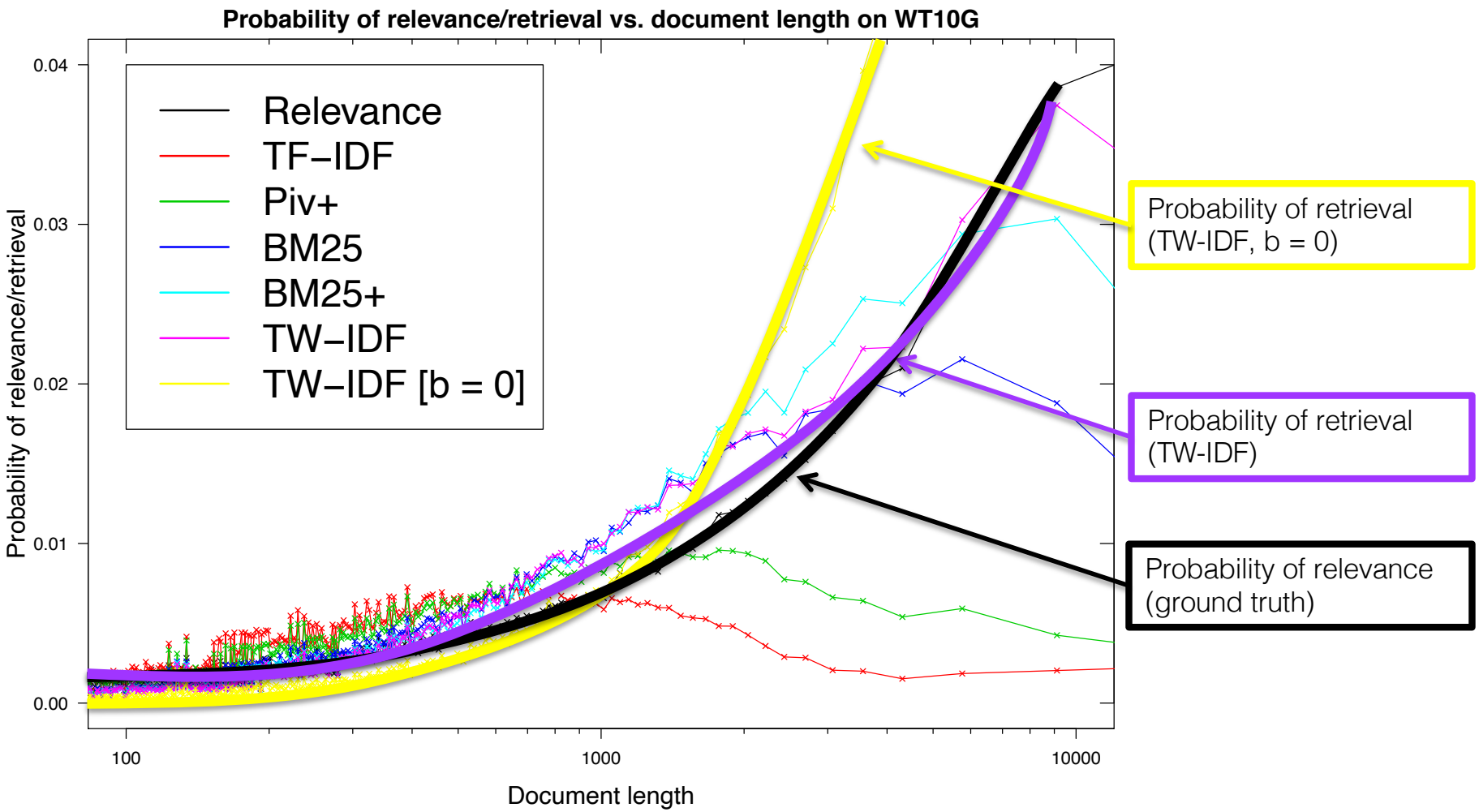
Graph-based Ad Hoc IR

- Evaluation in terms of:
  - Mean Average Precisio
  - Precision@10
  - Probability of relevanc
    vs. probability of retrie



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



Outline

- Graph-of-words
- **Keyword extraction**

Single Document Keyword Extraction

Keywords are used everywhere

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

Applications are numerous

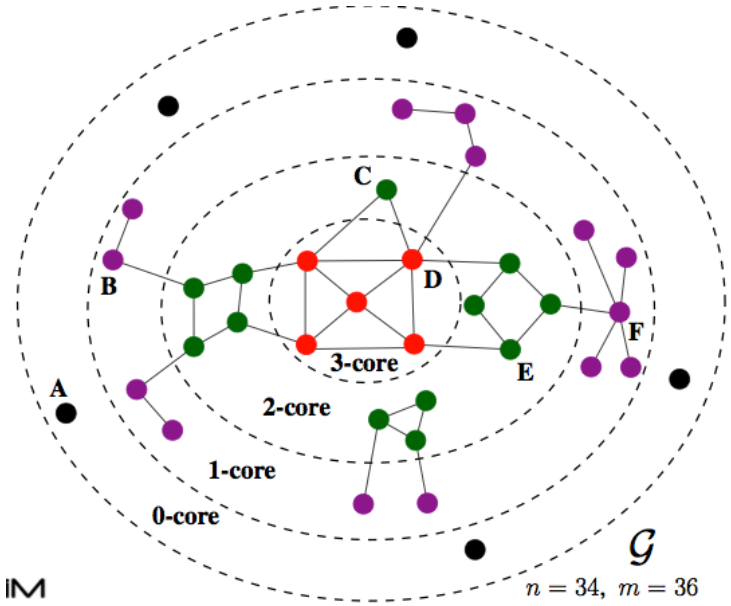
- **Summarization** (to get a gist of the content of a document)
- **Information filtering** (to select specific documents of interest)
- **Indexing** (to answer keyword-based queries)
- **Query expansion** (using additional keywords from top results)

Graph-based Keyword Extraction (1/2)

Existing graph-based keyword extractors:

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

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

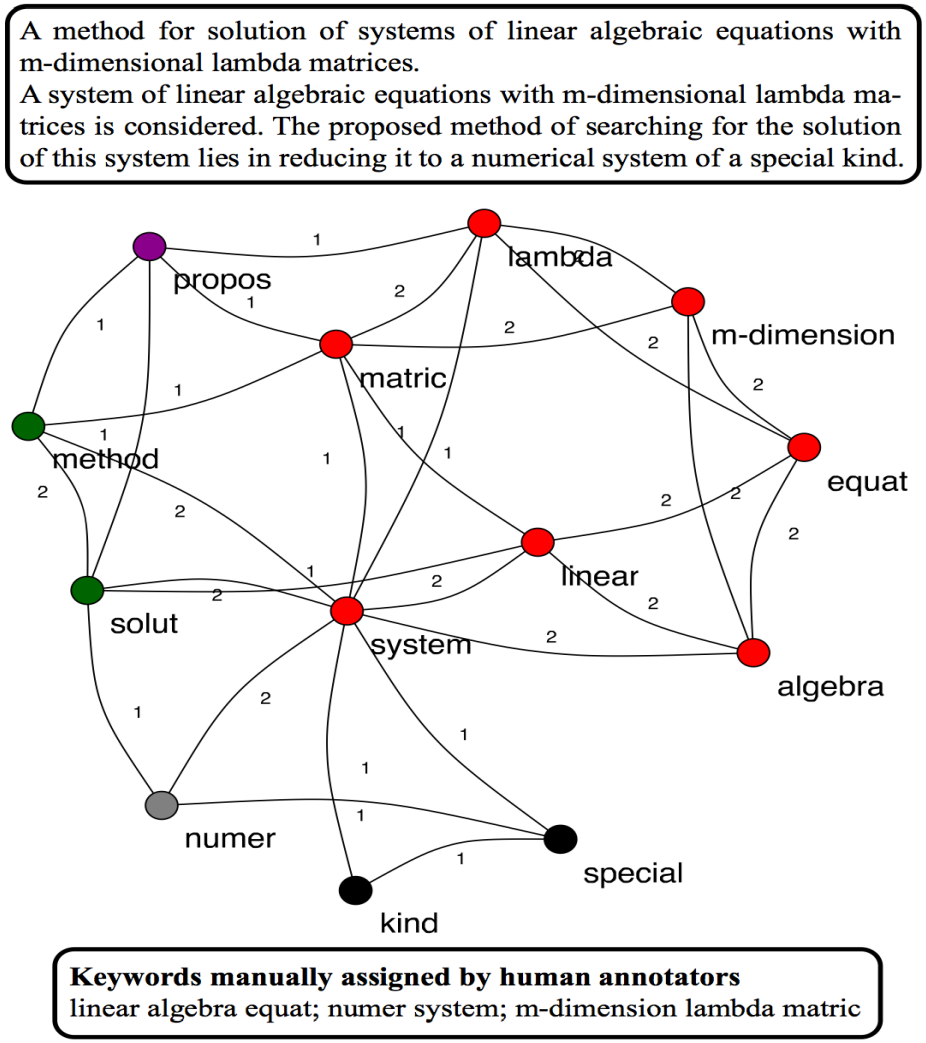


k-core decomposition of the graph

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

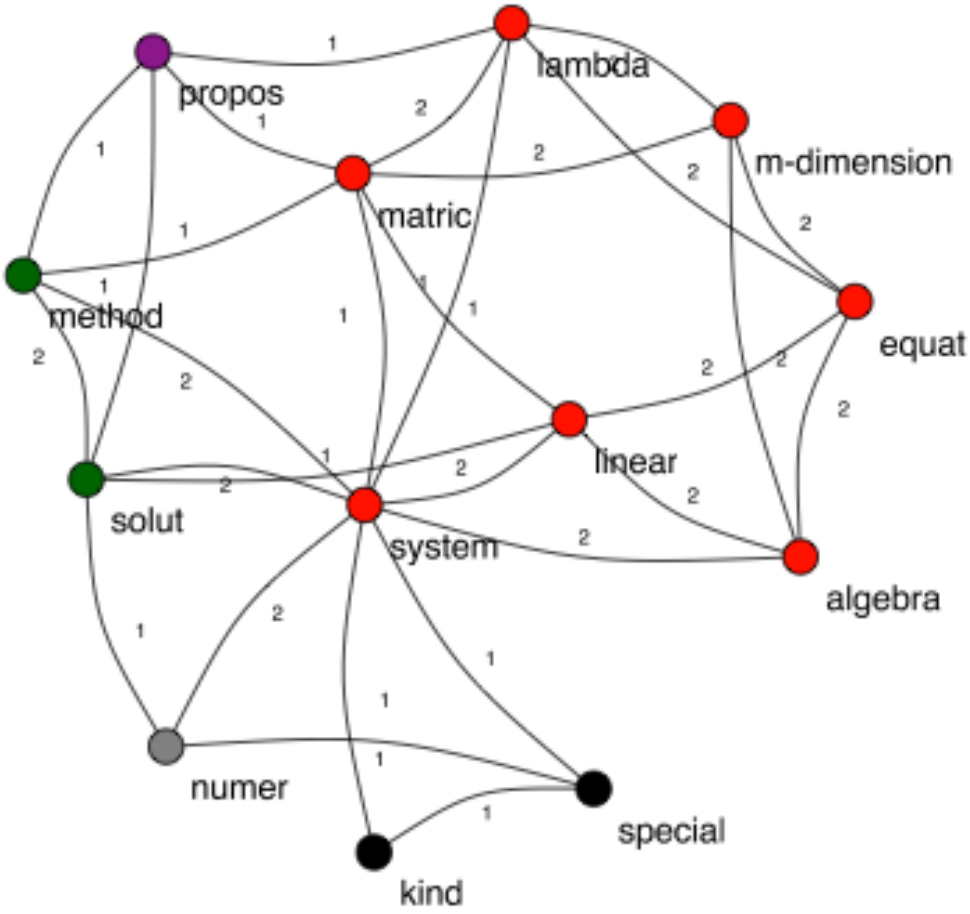
Graph-based Keyword Extraction (2/2)

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



[Rousseau and Vazirgiannis, ECIR '15]

PageRank vs. k-core



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

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

Keywords are not Unigrams

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

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

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

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

How Many Keywords?

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

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

k-cores are adaptive

Datasets

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

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

All approaches are unsupervised and single-document

Models and Baseline Methods

Graph-of-words:

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

Keyword extractors:

- PageRank
- HITS (authority scores only)
- k-core
- Weighted k-core

Top 33% or top 15% keywords

Main core

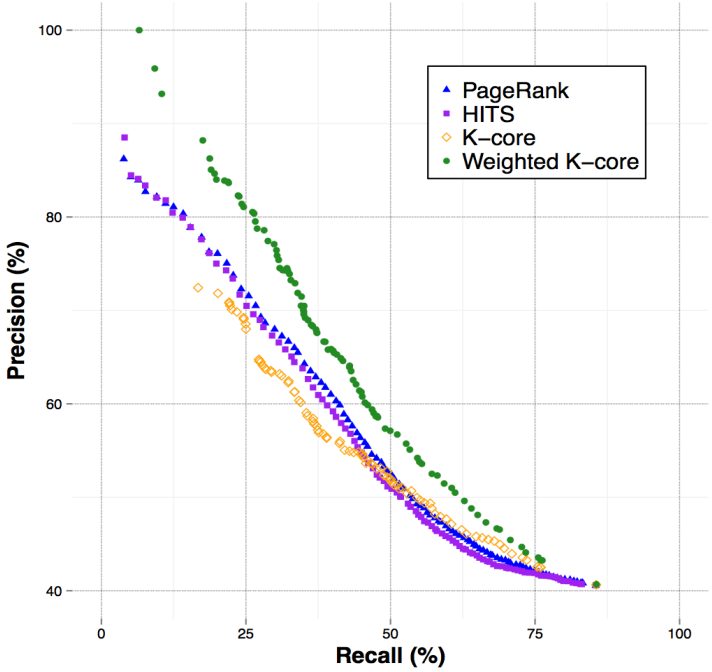
Performance Evaluation

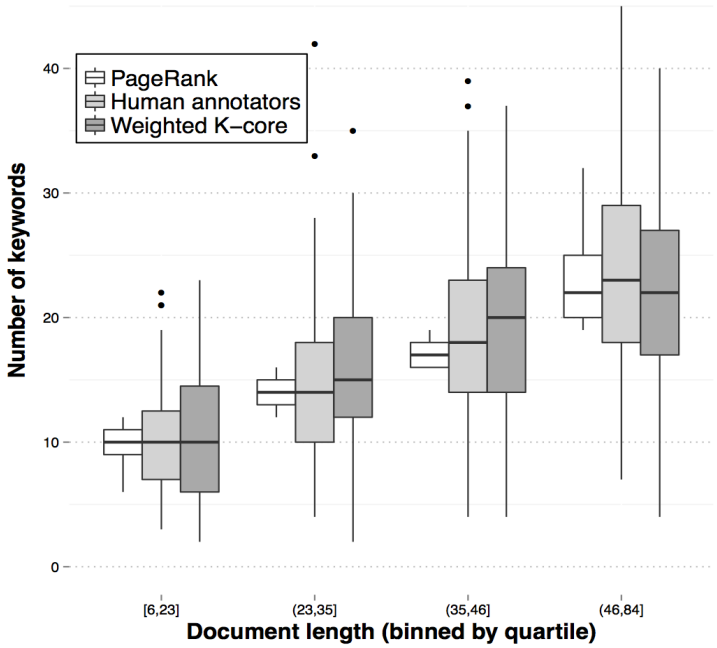
Precision

Recall

F1-score

Precision/recall





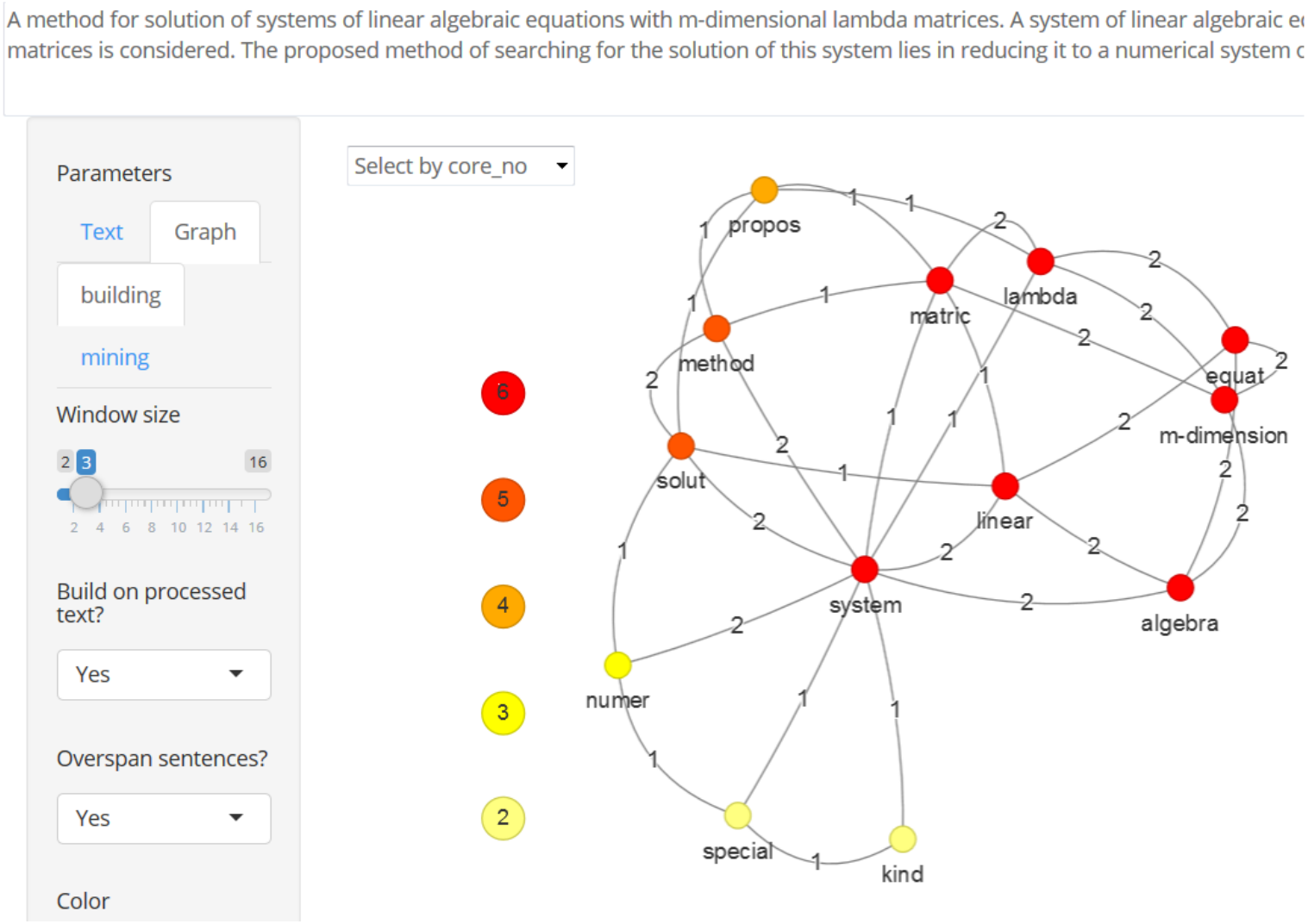
Graph	Dataset	Macro-averaged precision (%)				Macro-averaged recall (%)				Macro-averaged F1-score (%)			
		PageRank	HITS	K-core	WK-core	PageRank	HITS	K-core	WK-core	PageRank	HITS	K-core	WK-core
undirected edges	Hulth2003	58.94	57.86	46.52	**61.24***	42.19	41.80	**62.51***	50.32*	47.32	46.62	49.06*	**51.92***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

Example – ECIR'15 Paper

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

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

GoWvis Visualization Tool



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

[Tixier et al., ACL '16]

References (1)

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

References (2)

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

Outline

- TF/IDF - Text Retrieval
- Graph Based IR and keyword extraction
- **Attention in deep learning for NLP**
- RNN for machine translation

Attention is ubiquitous in DL today

Image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

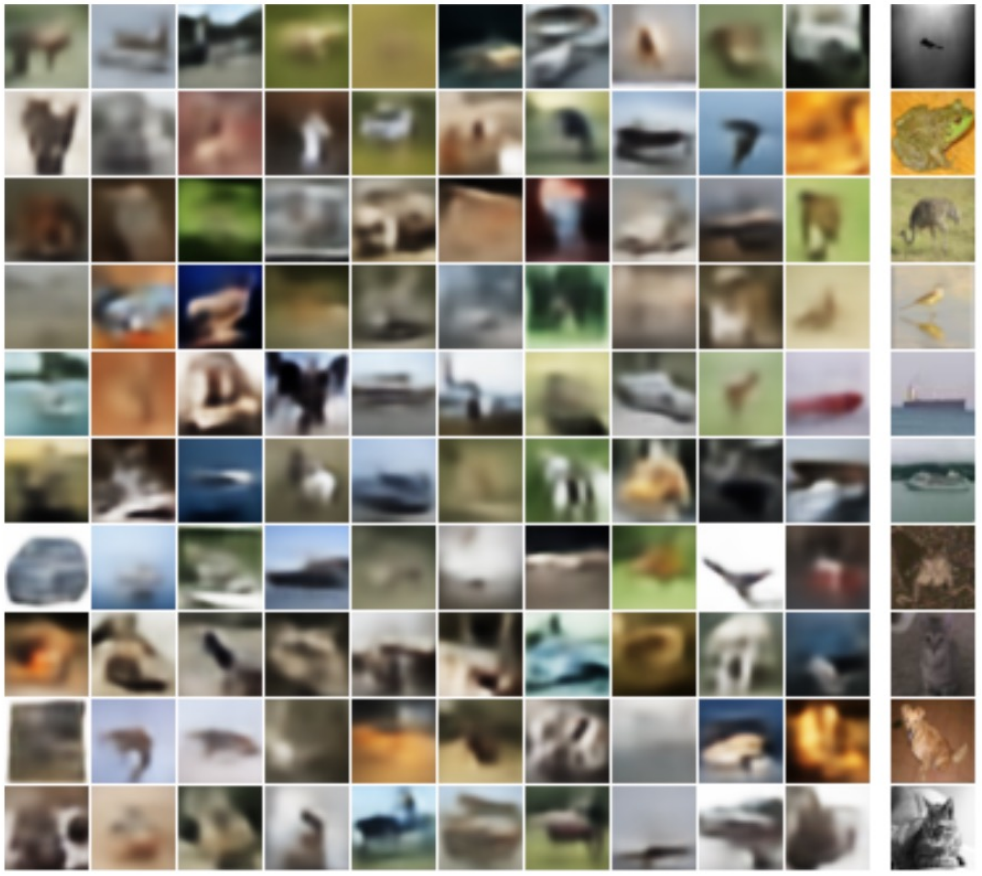
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Show, attend and tell: Neural image caption generation with visual attention (Xu et al. 2015)

Attention is ubiquitous in DL today

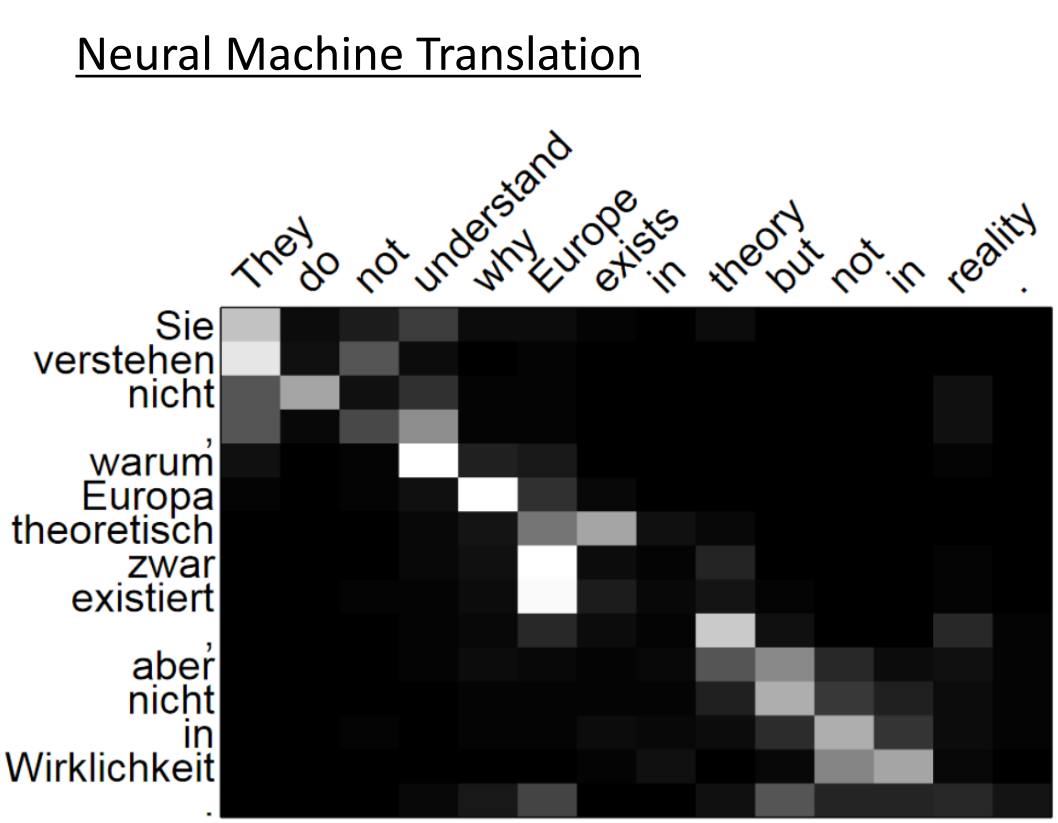
Image generation



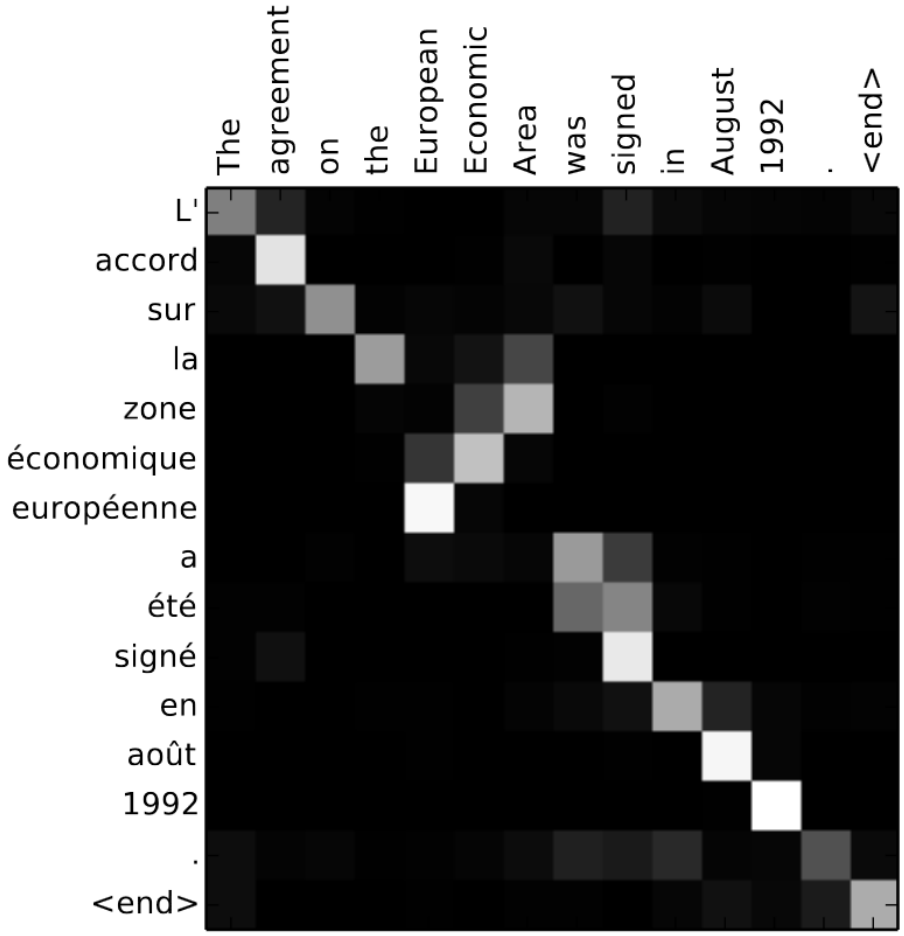
DRAW: A recurrent neural network for image generation (Gregor et al. 2015)

Attention is ubiquitous in DL today

Neural Machine Translation



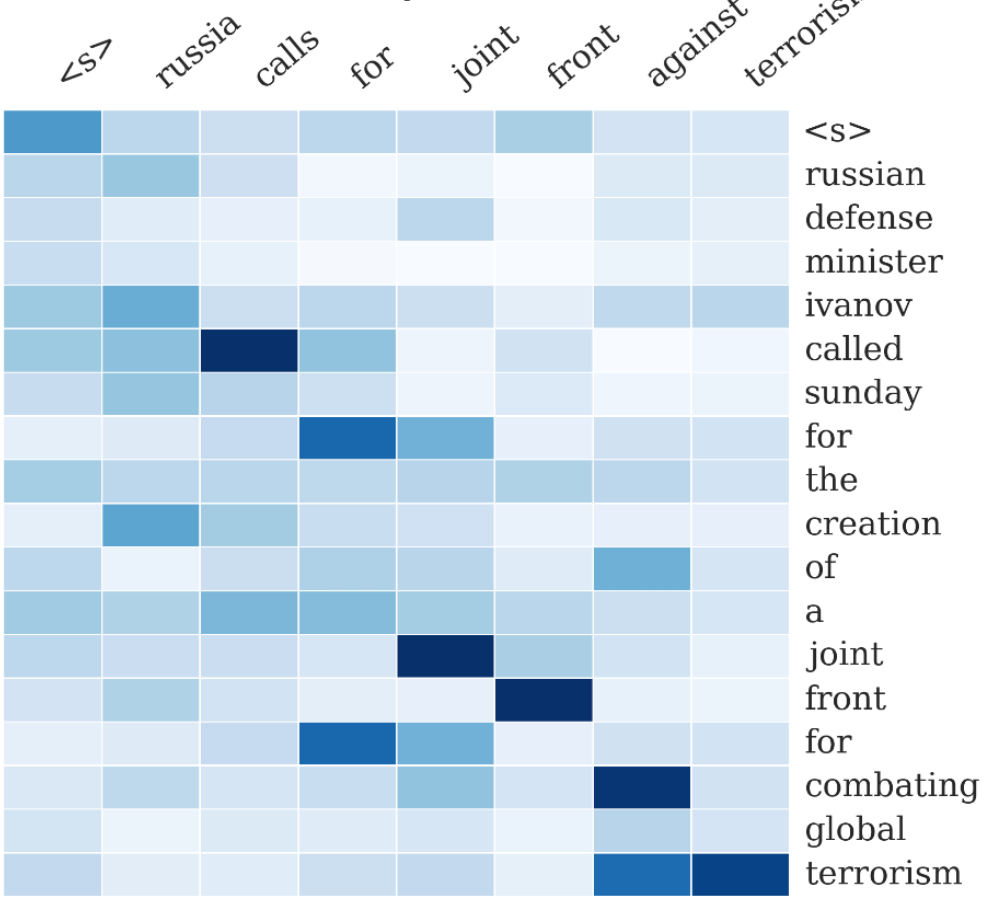
Effective approaches to attention-based neural machine translation (Luong et al. 2015)



Neural machine translation by jointly learning to align and translate (Bahdanau et al. 2014)

Attention is ubiquitous in DL today

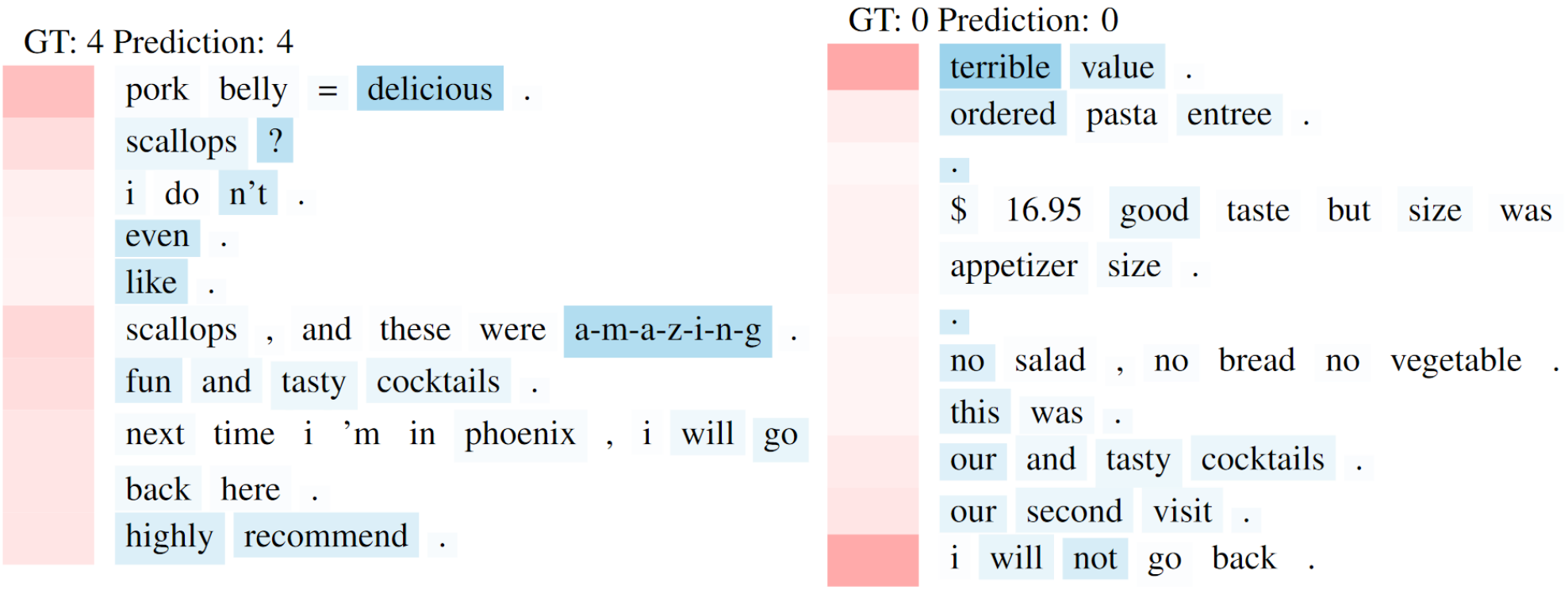
Abstractive summarization



A neural attention model for abstractive sentence summarization (Rush et al. 2015)

Attention is ubiquitous in DL today

Sentiment analysis



Hierarchical attention networks for document classification (Yang et al. 2016)

Encoder-Decoder Architectures

General idea:

encoder **decoder**

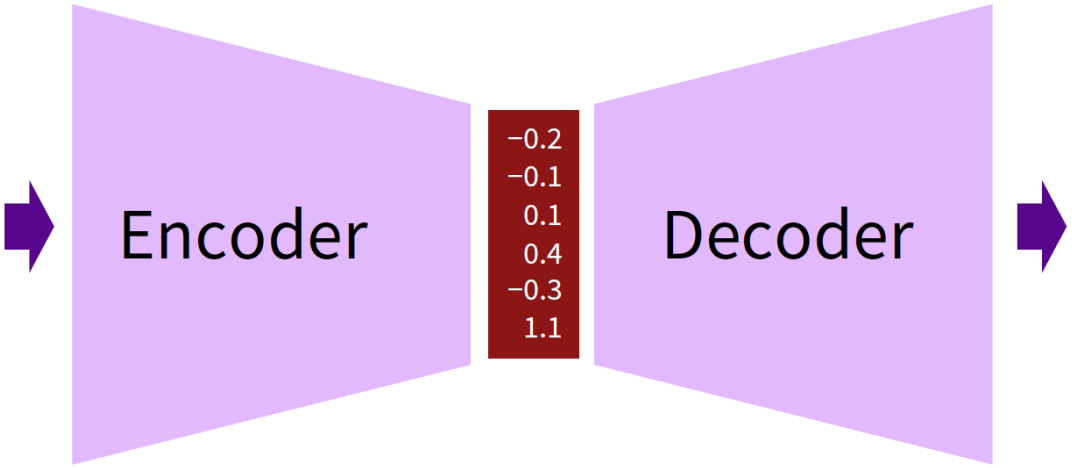
Input -> embedding -> Output

- image
- document
- source language
- …

- caption
- summary
- target language
- …

Known as *sequence-to-sequence* (seq2seq) when input and output are sequences (e.g., NLP applications)

The full architecture is differentiable => end-to-end training



What is attention?

Objective

- Traditional models (e.g., encoder) having to embed the input into a single fixed-length vector (lossy).
- Alternatively information can be kept and stored into multiple vectors.
- information can be retrieved later on (e.g., by the decoder). (Bahdanau et al. 2014)

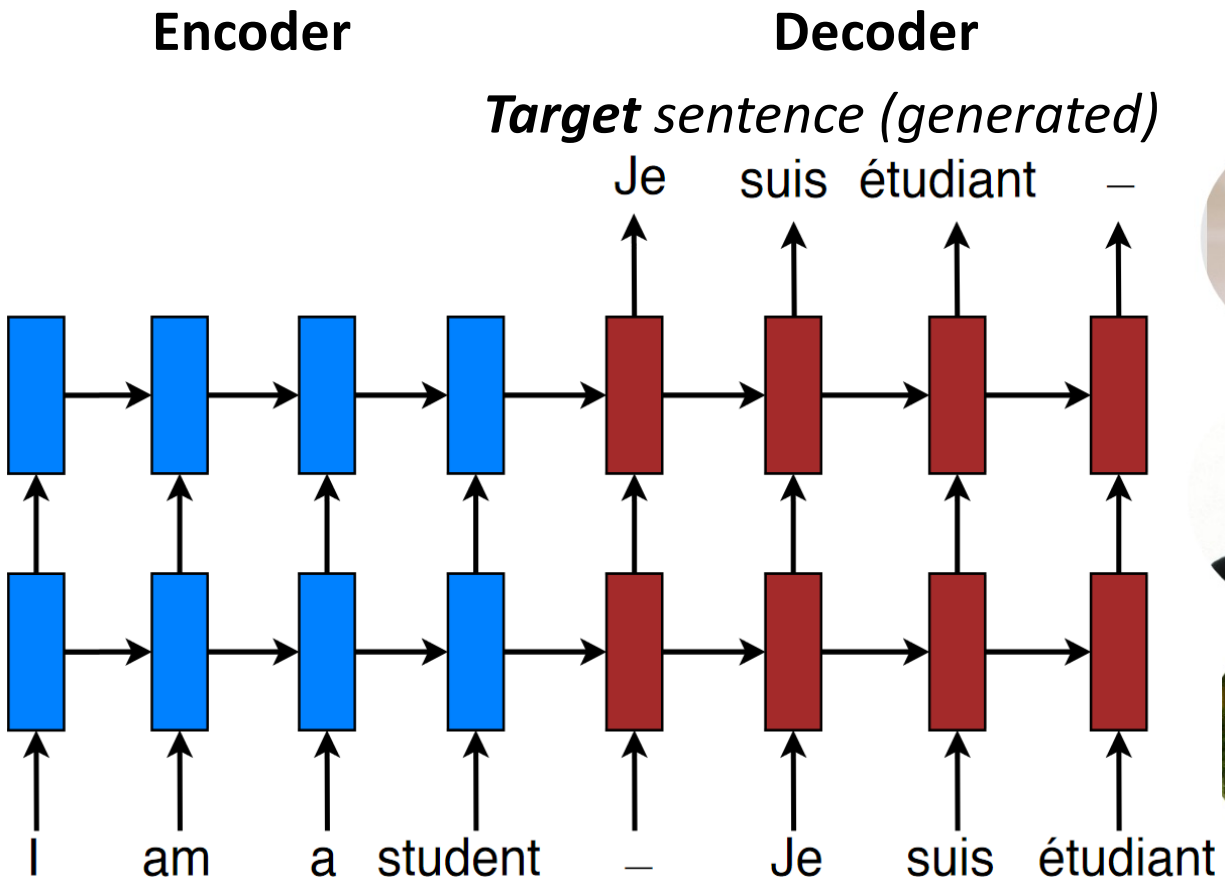
Quick history:

- developed in the context of **encoder-decoder** architectures for neural machine translation (Bahdanau et al. 2014)
- rapidly applied to naturally related tasks like image captioning (Xu et al. 2015) and summarization (*Luong et al. 2015*)
- also proposed for encoders only, e.g. for text classification (Yang et al. 2015) and representation learning (Conneau et al. 2017).
  Known as *self or inner attention* in such cases.

Outline

- TF/IDF - Text Retrieval
- Graph Based IR and keyword extraction
- Attention in deep learning for NLP
- **RNN for machine translation**

Encoder-Decoder for Neural Machine Translation



Minh-Thang Luong

Research Scientist at Google
Verified email at google.com - Homepage
Deep Learning Natural Language Processing

Hieu Pham

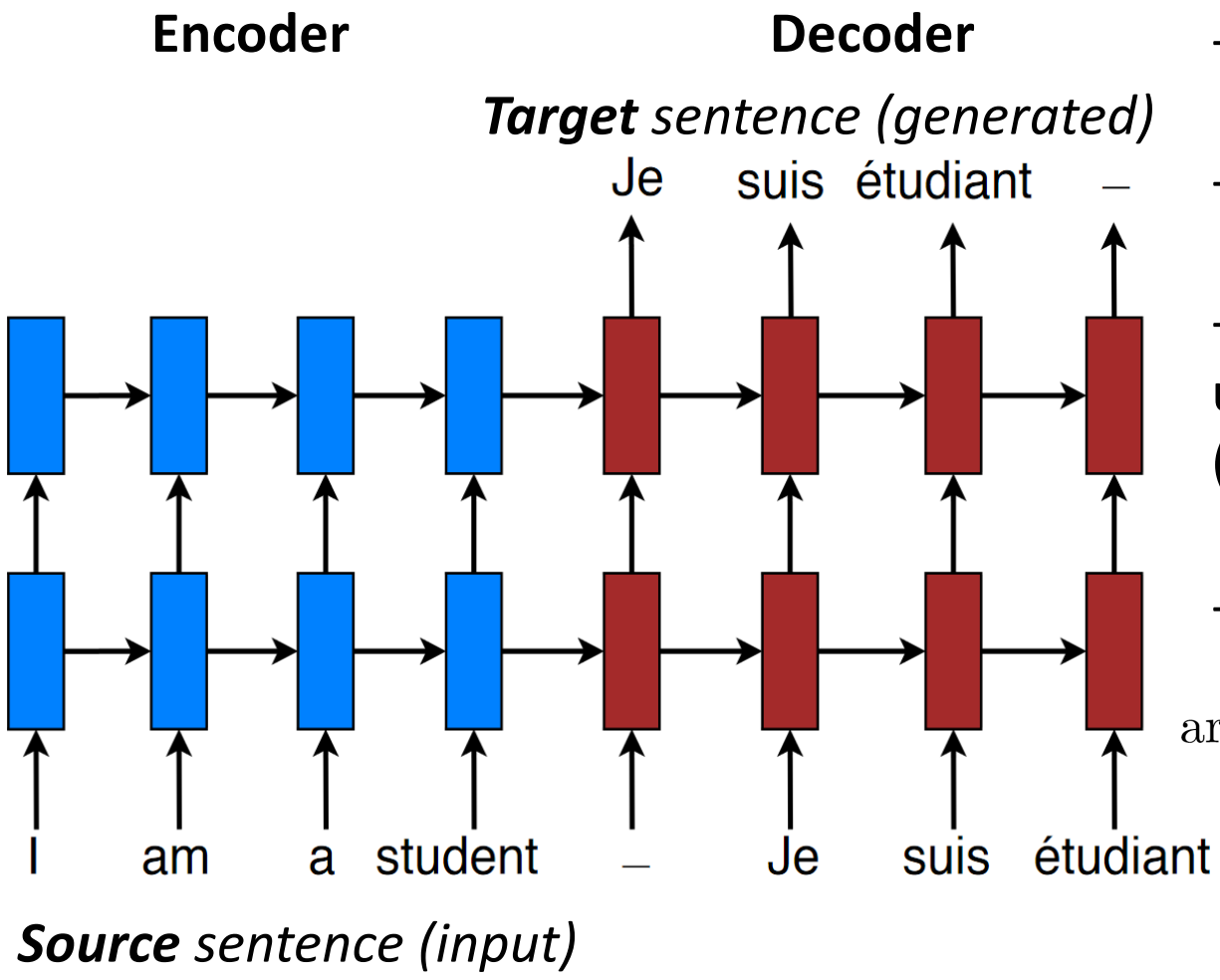
Carnegie Mellon University, Google Brain
Verified email at google.com
Machine Learning

Christopher D Manning

Professor of Computer Science and Linguistics
Verified email at stanford.edu - Homepage
Natural Language Processing

Effective approaches to Attention-Based Neural Machine Translation (2015)

Encoder-Decoder for Neural Machine Translation



- source: $\left(x_1, \ldots, x_{T_x}\right)$

- target: $\left(y_1, \ldots, y_{T_y}\right)$

- Both encoder & decoder are unidirectional deep RNNs (a.k.a. *stacking* RNNs)

- Training objective:

$$\text{argmax}_\theta \left\{ \sum_{(x,y)\in \text{corpus}} \log\, p(y|x;\theta) \right\}$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Encoder: usually: CNN, stacking RNN* with LSTM or GRU units…

* unidirectional (Luong et al. 2015) or bidirectional (Bahdanau et al. 2014).

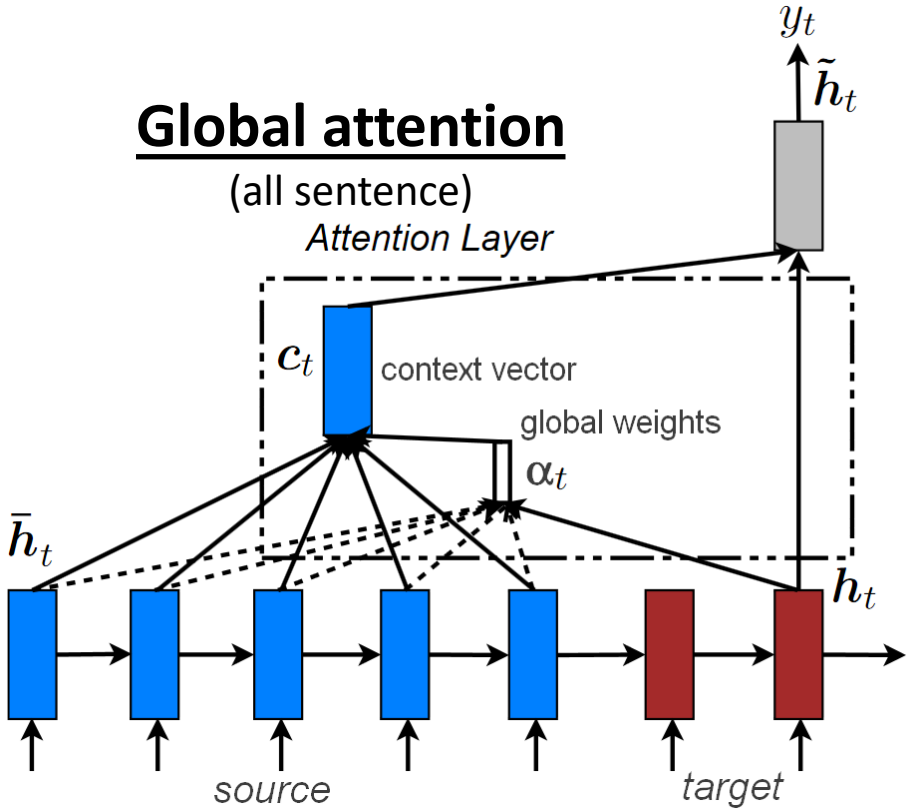
Decoder: unidirectional *RNN* (well suited to text generation), best if deep.

Generates the target sentence $\left(y_1, \cdots, y_{T_y}\right)$ one word at a time:

$$P\left[y_t | \{y_1, \ldots, y_{t-1}\}, c_t\right] = \text{softmax}\left(W_s \tilde{h}_t\right)$$

Luong et al. (2015)

Encoder-**Decoder** for Neural Machine Translation



$$P\left[y_t|\{y_1,...,y_{t-1}\},c_t\right] = \text{softmax}\left(W_s\tilde{h}_t\right)$$

attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c\left[c_t;h_t\right]\right)$$

context vector

decoder hidden state

$$c_t = \sum_{i=1}^{T_x} \alpha_{t,i}\bar{h}_i$$

i^{th} encoder hidden state

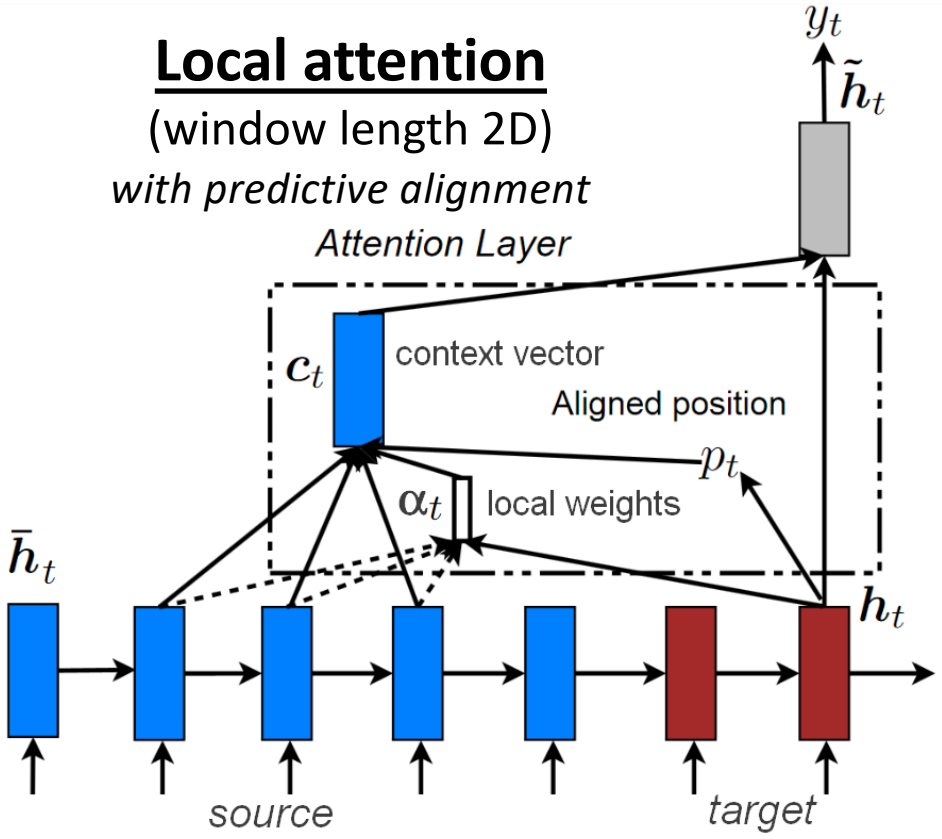
alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\text{score}\left(h_t,\bar{h}_i\right)\right)}{\sum_{i'=1}^{T_x}\exp\left(\text{score}\left(h_t,\bar{h}_{i'}\right)\right)}$$

$$\text{score}(h_t,\bar{h}_i) = h_t^\top \bar{h}_i$$

Luong et al. (2015)

Encoder-**Decoder** for Neural Machine Translation



$$P\left[y_t|\{y_1,...,y_{t-1}\},c_t\right] = \text{softmax}\left(W_s\tilde{h}_t\right)$$

attentional hidden state

$$\tilde{h}_t = \tanh\left(W_c\left[c_t; h_t\right]\right)$$

decoder hidden state

$$p_t = T_x \cdot \sigma\left(v_p^\top \tanh\left(W_p h_t\right)\right)$$

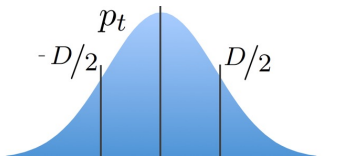
context vector

$$c_t = \sum_{i=p_t-D}^{p_t+D} \alpha_{t,i}\bar{h}_i$$

i[th] encoder hidden state

alignment vector

$$\alpha_{t,i} = \frac{\exp\left(\text{score}(h_t,\bar{h}_i)\right)}{\sum_{i'=p_t-D}^{p_t+D}\exp\left(\text{score}(h_t,\bar{h}_{i'})\right)}\exp\left(-\frac{(i-p_t)^2}{2(D/2)^2}\right)$$



$$\text{score}(h_t,\bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$$

Luong et al. (2015)

Encoder-Decoder for Neural Machine Translation

Facts:

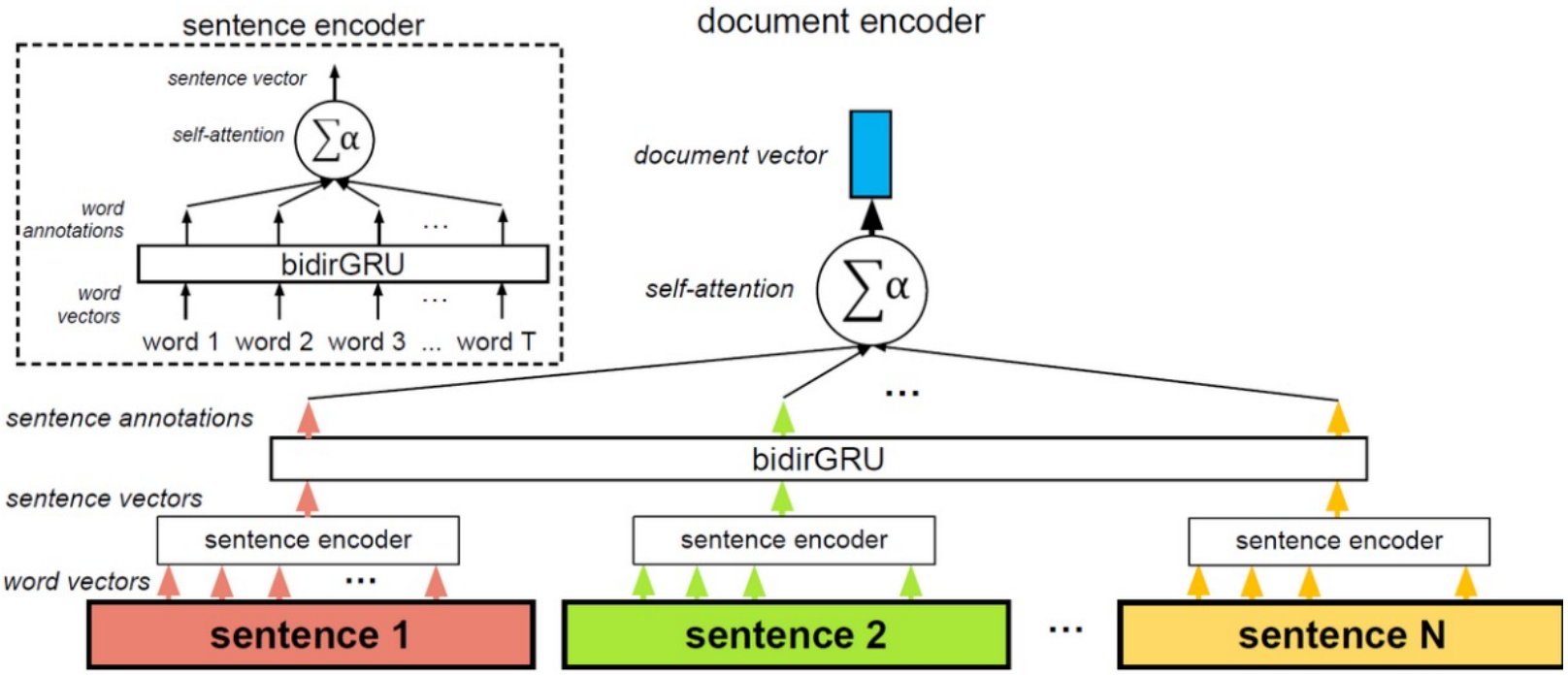
- Tested on the English <-> German task (WMT'14 dataset)
- 4.5M sentence pairs
- Encoder and decoder RNNs feature 4 layers of stacking and 1000-dimensional hidden states
- Window of size *D=10* for local attention
- Trained for 12 epochs (total of 7-10 days on a single GPU, at 1K target words/s)
- New state-of-the-art performance

Lessons learned:

- Local attention with predictive alignment gives better results than global attention
- Dot product ($\text{score}(h_t, \bar{h}_i) = h_t^\top \bar{h}_i$) works well for global attention
- The general formulation ($\text{score}(h_t, \bar{h}_i) = h_t^\top W_\alpha \bar{h}_i$) is better for local attention

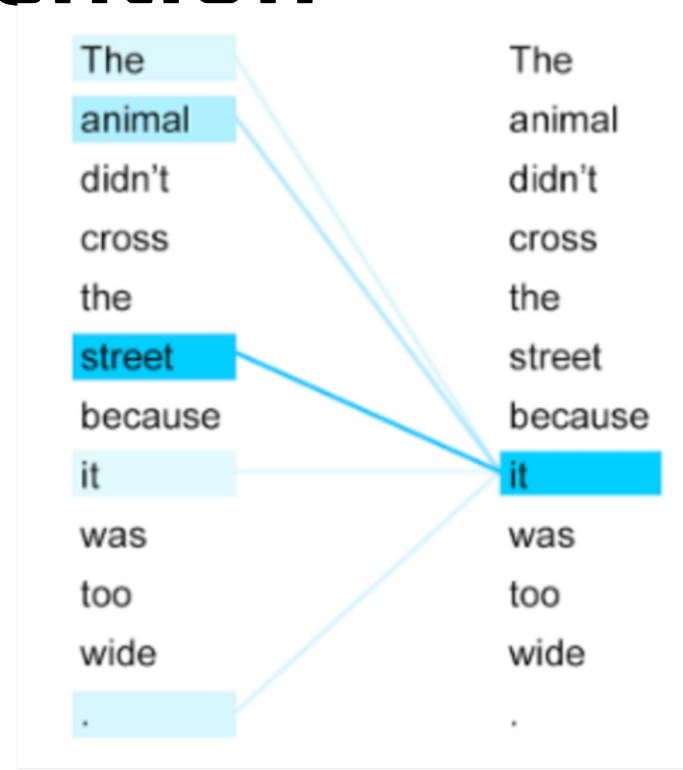
Luong et al. (2015)

Self-attention for RNN encoders



Self Attention

“Self-attention: mechanism relating different words of a single sequence in order to compute a representation of the sequence.”



http://jalammar.github.io/illustrated-transformer/

Self-attention for RNN encoders

- Input is a sentence $(x_1, \ldots, x_T)$
- We're only interested in getting an embedding s of the sentence for some downstream task (e.g., classification)

encoder hidden state

context vector

$$u_t = \tanh(W h_t)$$

$$\alpha_t = \frac{\exp(\text{score}(u_t, u))}{\sum_{t'=1}^{T} \exp(\text{score}(u_{t'}, u))}$$

$$s = \sum_{t=1}^{T} \alpha_t h_t$$

$$\text{Where score}(u_t, u) = u_t^\top u$$

The same process can be repeated over the sentence vectors s -> **hierarchical attention**