Evaluating the Impact of Word Embeddings on Similarity Scoring for Practical Information Retrieval

Lukas Galke Ahmed Saleh Ansgar Scherp

Leibniz Information Centre for Economics

Kiel University

INFORMATIK, 29 Sep 2017

Motivation and Research Question

Motivation

- Word embeddings regarded as the main cause for NLP breakout in the past few years (Goth 2016)
- can be employed in various natural language processing tasks such as classification (Balikas and Amini 2016), clustering (Kusner et al. 2015), word analogies (Mikolov et al. 2013), language modelling ...
- Information Retrieval is quite different from these tasks, so employment of word embeddings is challenging

Research question

Which embedding-based techniques are suitable for similarity scoring in practical information retrieval?

Related Approaches

Paragraph Vectors (Doc2Vec)

Explicitly learn document vectors in a similar fashion to word vectors (Le and Mikolov 2014). One artificial token per paragraph (or document).

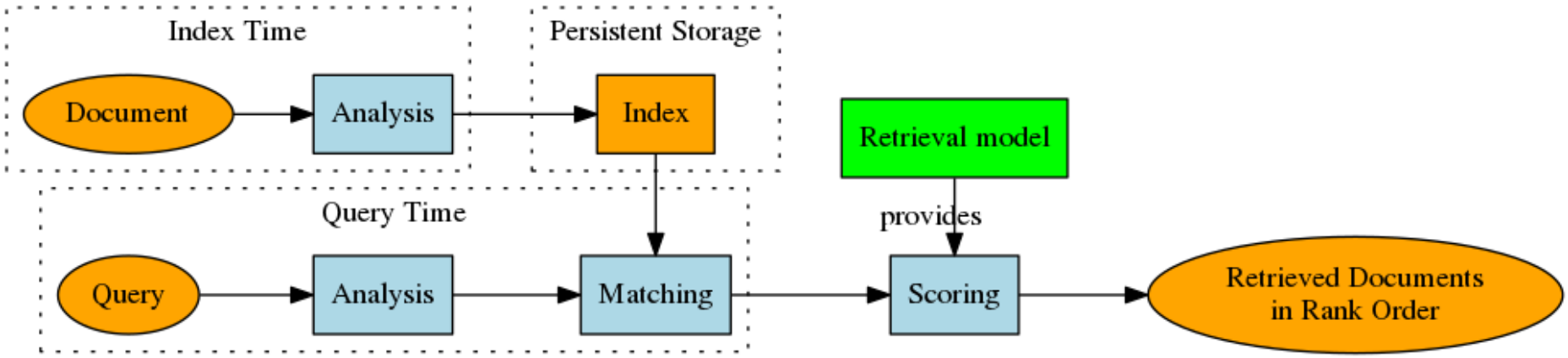
Word Mover's Distance (WMD)

Compute Word Mover's Distance to solve a constrained optimization problem to compute document similarity (Kusner et al. 2015). Minimize the cost of *moving* the words of one document to the words of the other document.

Embedding-based Query Language Models

Embedding-based query expansion and embedding-based pseudo relevance feedback (Zamani and Croft 2016). The query is expanded by nearby words with respect to the embedding.

Information Retrieval



Information Retrieval

Given a query, retrieve the k most relevant (to the query) documents from a corpus in rank order.

TF-IDF Retrieval Model

Term frequencies

TF(w, d) is the number of occurrences of word w in document d.

TF-IDF Retrieval Model

Term frequencies

TF(w, d) is the number of occurrences of word w in document d.

Inverse Document Frequency

Words that occur in a lot of documents are discounted (assuming they carry less discriminative information): $\mathrm{IDF}(w, D) = \log \frac{|D|}{|\{d \in D | w \in d\}|}$

TF-IDF Retrieval Model

Term frequencies

$\mathrm{TF}(w, d)$ is the number of occurrences of word w in document d.

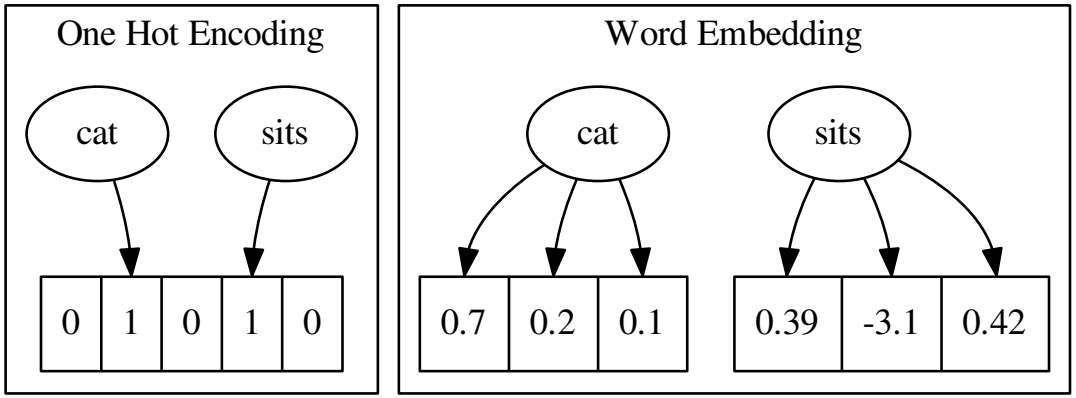
Inverse Document Frequency

Words that occur in a lot of documents are discounted (assuming they carry less discriminative information): $\mathrm{IDF}(w, D) = \log \frac{|D|}{|\{d \in D | w \in d\}|}$

Retrieval Model

- Transform corpus of documents d into TF-IDF representation.
- Compute TF-IDF representation of the query q.
- Rank matching documents by descending cosine similarity $\cos_{\mathsf{sim}}(q, d) = \frac{q \cdot d}{\|q\| \cdot \|d\|}$.

Word Embedding



Word Embedding

Low-dimensional (compared to vocabulary size) distributed representation, that captures *semantic* and *syntactic* relations of the words. Key principle: *Similar words should have a similar representation.*

Word Vector Arithmetic

Addition of word vectors and its nearest neighbors in the word embedding[1].

Expression	Nearest tokens
Czech + Currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg

[1] Extracted from Mikolov's talk at NIPS 2013

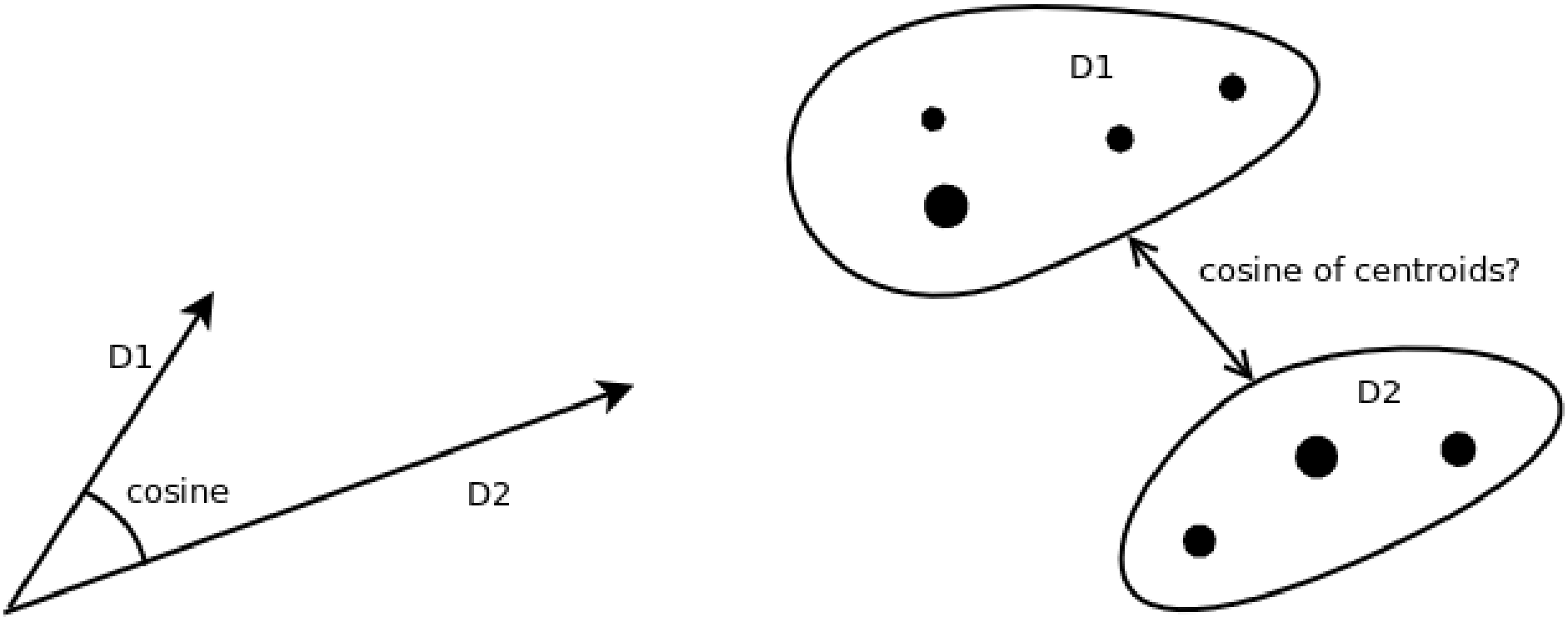
Skip-Gram Negative Sampling (Mikolov et al. 2013)

Given a stream of words (tokens) s over vocabulary V and a context size k, learn word embedding W.

- Let w_T be target word with context $C = \{s_{T-k}, \ldots, s_{T-1}, s_{T+1}, \ldots, s_{T+k}\}$ (**skip-gram**)
- Look up word vector $W[s_T]$ for target word s_T
- Predict via logistic regression from word vector $W[s_T], W[x]$ with:
  - positive examples: x context words C
  - negative examples: x sampled from $V \setminus C$ (**negative sampling**)
- Update word vector $W[s_T]$ (via back-propagation)
- Repeat with next word $T = T + 1$

Document Representation

How to employ word embeddings for information retrieval?



Bag-of-words (left) vs distributed representations (right)

Embedded Retrieval Models

Word Centroid Similarity (WCS)

Aggregate word vectors to their centroid for both the documents and the query and compute cosine similarity between the centroids.

Word vector centroids $C = TF \cdot W$

Given query q in one-hot representation, compute

Word Centroid Similarity $WCS(q, i) = \frac{(q^T \cdot W) \cdot C_i}{\|q_T \cdot W\| \cdot \|C_i\|}$

Embedded Retrieval Models

Word Centroid Similarity (WCS)

Aggregate word vectors to their centroid for both the documents and the query and compute cosine similarity between the centroids.

Word vector centroids $C = TF \cdot W$

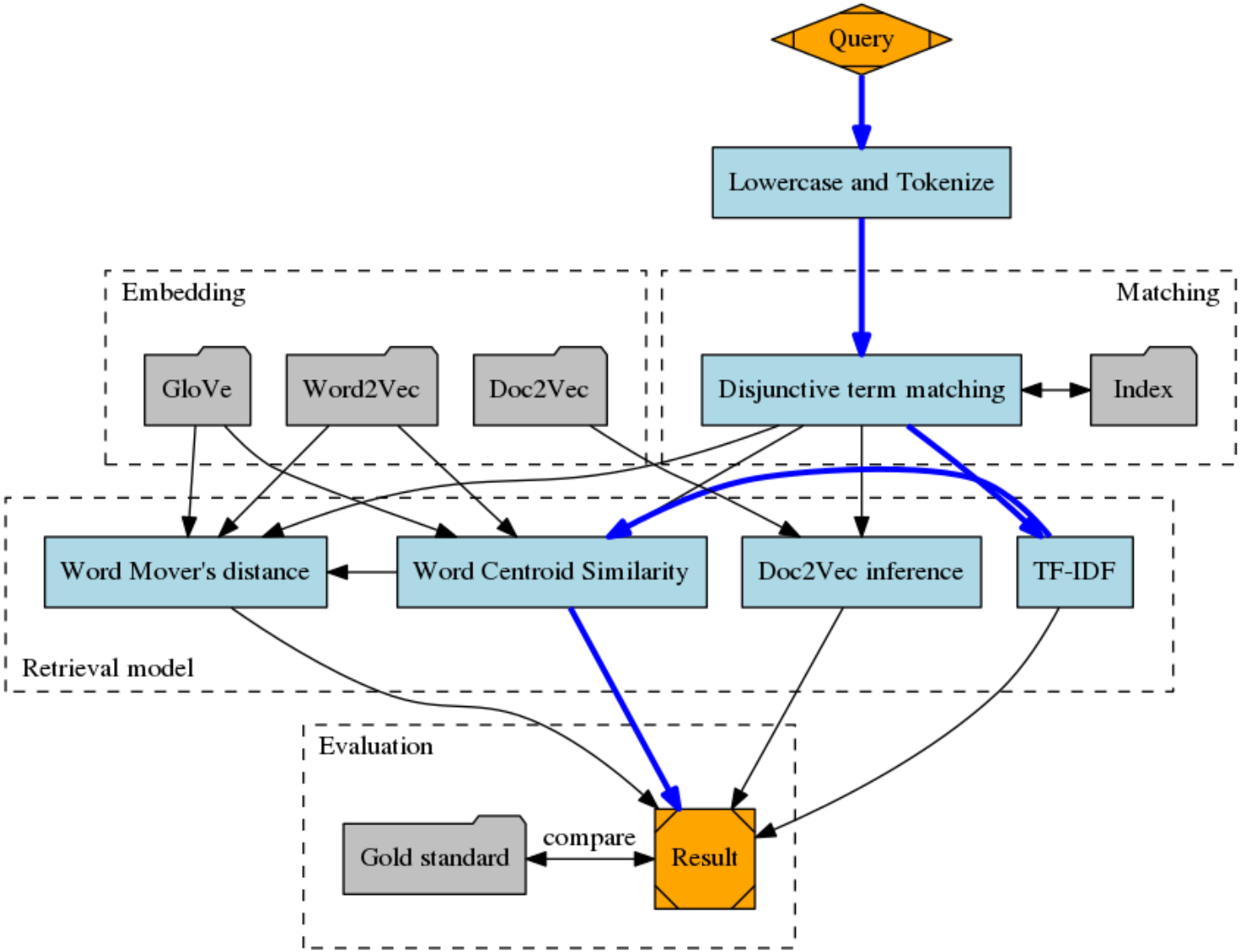
Given query q in one-hot representation, compute

Word Centroid Similarity $WCS(q, i) = \frac{(q^T \cdot W) \cdot C_i}{\|q_T \cdot W\| \cdot \|C_i\|}$

IDF re-weighted Word Centroid Similarity (IWCS)

IDF re-weighted aggregation of word vectors $C = TF \cdot IDF \cdot W$

Experiments



Results

- Ground truth: Relevancy provided by human annotators.
- Evaluate mean average precision of top 20 documents ($MAP@20$)
- Datasets: Titles of NTCIR2, Economics, Reuters

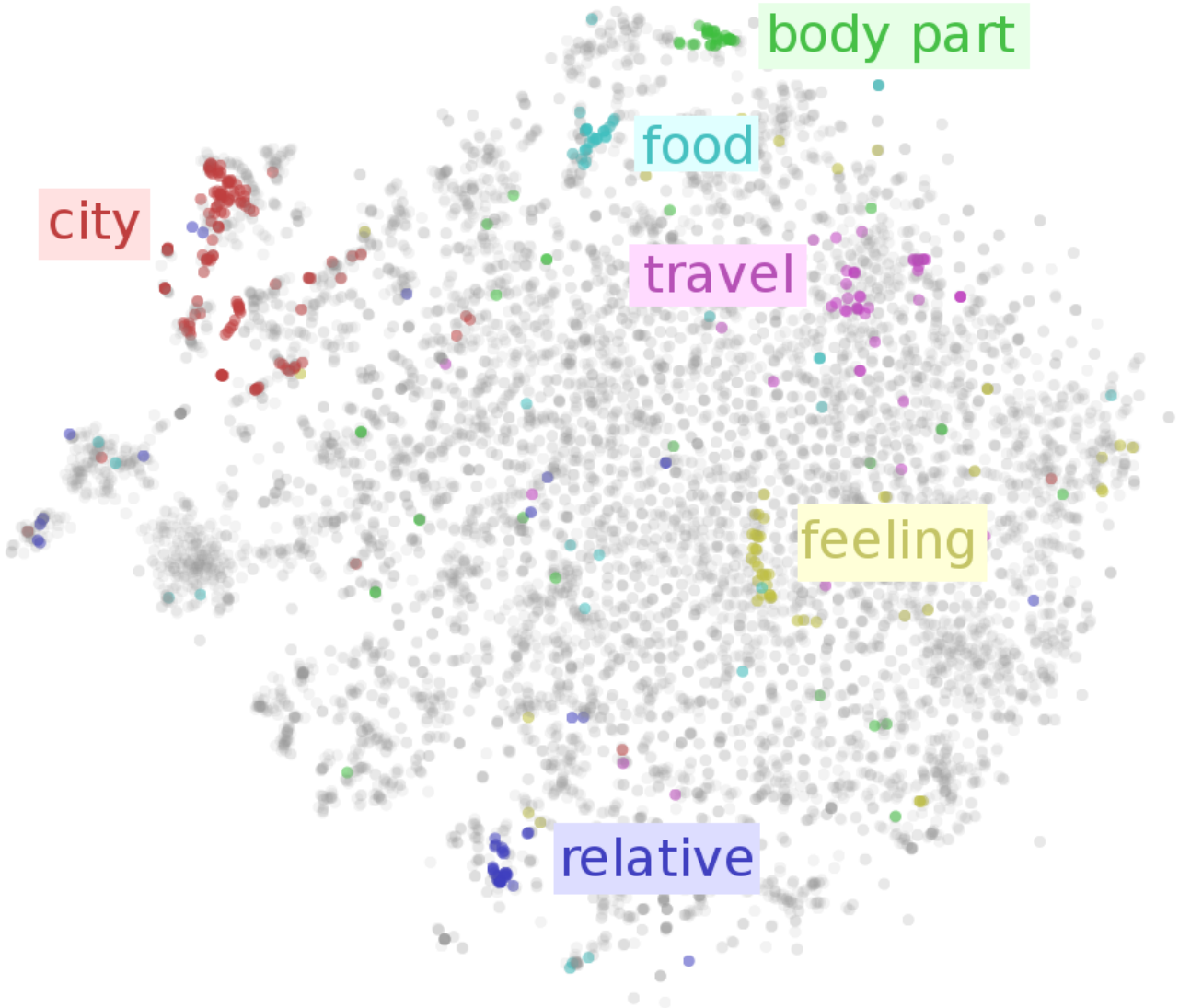
Model	NTCIR2	Economics	Reuters
TF-IDF (baseline)	.40	**.37**	.52
WCS	.30	.36	.54
IWCS	**.41**	**.37**	**.60**
IWCS-WMD	.40	.32	.54
Doc2Vec	.24	.30	.48
...			

Conclusion

- Word embeddings can be successfully employed in practical IR
- IWCS is competitive to the TF-IDF baseline
- IDF weighting improves the performance of WCS by 11%
- IWCS outperforms the TF-IDF baseline by 15% on Reuters (news domain)
- Code to reproduce the experiments is available at github.com/lgalke/vec4ir.

MOVING is funded by the EU Horizon 2020 Programme under the project number INSO-4-2015: 693092

Discussion



Data Sets and Embeddings

Data set properties

Data set	Documents	Topics	relevant per topic
NTCIR2	135k	49	43.6 (48.8)
Econ62k	62k	4518	72.98 (328.65)
Reuters	100k	102	3,143 (6,316)

Word Embedding properties

Embedding	Tokens	Vocab	Case	Dim	Training
GoogleNews	3B	3M	cased	300	Word2Vec
CommonCrawl	840B	2.2M	cased	300	GloVe

Preprocessing in Detail

Matching and TFIDF

- Token Regexp: `\w\w+`
- English stop words removed

Word2Vec

- Token Regexp: `\w\w*`
- English stop words removed

GloVe

- Punctuation separated by white-space
- Token Regexp: `\S+` (everything but white-space)
- No stop word removal

Out of Vocabulary Statistics

Embedding	Data set	Field	OOV ratio
GoogleNews	NTCIR2	Title	7.4%
		Abstract	7.3%
	Econ62k	Title	2.9%
		Full-Text	14.1%
CommonCrawl	NTCIR2	Title	5.1%
		Abstract	3.5%
	Econ62k	Title	1.2%
		Full-Text	5.2%

Metrics in Detail

Let r be relevance scores in rank order as retrieved. Nonzero indicating true positive and zero false positive. For each metric, the scores are averaged over the queries.

Reciprocal Rank

$\mathrm{RR}(r,k) = \frac{1}{\min\{i|r_i>0\}}$ if $\exists i : r_i > 0$ else 0.

Average Precision

$\mathrm{Precision}(r,k) = \frac{|\{r_i \in r | r_i > 0\}|}{|k|}$,

$\mathrm{AP}(r,k) = \frac{1}{|r|} \sum_{i=1}^{k} Precision((r_1, \ldots, r_i), i)$

Normalised Discounted Cumulative Gain

$\mathrm{DCG}(r,k) = r_1 + \sum_{i=2}^{k} \frac{r_i}{\log_2 i}$, $\mathrm{nDCG}(r,k) = \frac{\mathrm{DCG}(r,k)}{\mathrm{IDCG_{q,k}}}$

where IDCG is the optimal possible DCG value for a query (w.r.t gold standard)

References

Balikas, Georgios, and Massih-Reza Amini. 2016. "An Empirical Study on Large Scale Text Classification with Skip-Gram Embeddings." *CoRR* abs/1606.06623.

Goth, Gregory. 2016. "Deep or Shallow, NLP Is Breaking Out." *Commun. ACM* 59 (3): 13–16.

Kusner, Matt J., Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. "From Word Embeddings to Document Distances." In *ICML*, 37:957–66. JMLR Workshop and Conference Proceedings. JMLR.org.

Le, Quoc V., and Tomas Mikolov. 2014. "Distributed Representations of Sentences and Documents." In *ICML*, 32:1188–96. JMLR Workshop and Conference Proceedings. JMLR.org.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. "Distributed Representations of Words and Phrases and Their Compositionality." In *NIPS*, 3111–9.

Zamani, Hamed, and W. Bruce Croft. 2016. "Embedding-Based Query Language Models." In *ICTIR*, 147–56. ACM.