Text Retrieval Conference - Complex Answer Retrieval

Gaurav Patil
Department of Computer Science
University Of New Hampshire
Durham - NH - 03824
gp1026@wildcats.unh.edu

Shilpa Dhagat
Department of Computer Science
University Of New Hampshire
Durham - NH - 03824
sd1025@wildcats.unh.edu

Colin Etzel
Department of Computer Science
University Of New Hampshire
Durham - NH - 03824
cwg4@wildcats.unh.edu

Abstract—**Information retrieval systems were originally developed to help manage the huge scientific literature that has developed since the 1940s. Information retrieval is used today in many applications. Is used to search for documents, content thereof, document metadata within traditional relational databases or internet documents, more conveniently and decrease work to access information. Retrieved documents should be relevant to user's information need. Obvious examples include search engines like Google, Yahoo and Microsoft's Bing. Many problems in information retrieval can be viewed as prediction problem, i.e. to predict ranking scores or rating of web pages, documents, music songs etc. and learning the information desires and interests of users. Current retrieval systems provide good solutions towards phrase-level retrieval for simple fact and entity-centric needs. This track encourages research for answering more complex information needs with longer answers. Much like Wikipedia pages synthesize knowledge that is globally distributed, we envision systems that collect relevant information from an entire corpus, creating synthetically structured documents by collating retrieved results.**

1. Introduction

Complex Answer Retrieval is a track for TREC 2017. Its focus is to develop a system that can answer complex information needs by gathering relevant information from a given corpus. Given an article stub Q, retrieve for each of its sections Hi, a ranking of relevant passages. The passage S is taken from a provided passage corpus. We define a passage as relevant if the passage content should be mentioned in the knowledge article. Different degrees of "could be mentioned", "should be mentioned", or "must be mentioned" are represented through a PEGFB annotation scale during manual assessment.

We have implemented several ranking functions in our implementation to see which is the best overall. We test each function, add Query-Expansion, relevance feedback, stemming, stopping, and clustering, but although each ranking function use a different approach, the improvement on results is small and almost the same.

Given the paragraph collection and the queries, we produce the initial list of ranked results; the TREC run. To further improve the results, we perform re-ranking and relevance feedback on top k paragraphs.

As someone already familiar with Information Retrieval techniques, using clustering motivated us to solve problems that can't be easily solved with producing a ranked list of results common in information retrieval tasks so that information retrieval methods could work better. With a lax enough clustering metric to not lose significant amounts of relevant documents time could be saved by running an information retrieval algorithm on smaller clusters as opposed to the entire set of documents. On the other end of the pipeline, there was the possibility that clustering could be used to ensure documents ended up in the correct sections on the page level; perhaps putting the passages in the right sections could be achieved better by similarities the passages had to each other instead of to the often short and homogeneous query names in a typical page. The decision to try applying k-means clustering on the second scenario was chosen based on the suggestion to design a method as a pipeline and the organization of the TREC CAR data. If previous methods upstream could accurately map data to the correct page name, intuitively somewhat easier than mapping to the correct page and section name, there might be an inherent similarity structure sufficiently to divide the passages into the correct section name. Euclidean distance from TFIDF with a bag of words model seemed likely to be more effective as a distance metric than using it in a query search as there would be more words in common between paragraphs in the same section than between an individual paragraph and a section name, especially rare words not found in other sections.

Many of the runs seen at TREC are not simply the result of a simple ranking function, but of a pipeline of other processes (such as stemming, stopping, entity-linking, relevance feedback, clustering and so on). Current retrieval systems provide good solution towards phrase-level retrieval for simple facts. Recent work on search engine ranking functions report improvements on BM25. Our contributions are:

- Implementing retrieval algorithms like BM25+, Improved TF-IDF and LM-DS [1] that rank documents significantly better when compared to baseline BM25 [10,11].
- Pre-processing the query and document text by applying text-processing concepts like removal of common words, upper case to lower, removal of special symbols, word stemming, work ranking.
- Expanding query and paragraphs using entity linking and query using Pseudo Relevance Feedback.
- Re-ranking top n results using an alternate retrieval algorithm.
- Building index for documents and queries.
- Clustering for document grouping, ranking the passages with cosine similarity for evaluation purposes.

2. RELATED WORK

2.1. Retrieval Methods

Over time, many different types of retrieval models have been proposed and tested. We have implemented the classic probabilistic retrieval model (Okapi BM25), Improved methods to BM25 [1] and the recently proposed language modeling approach (Dirichlet prior smoothing). BM25 is often used as a baseline, and we do the same here. Problematically, many of the implementations of BM25 are different and comparisons are to BM25-like functions.

More recently, Trotman et al. [1] examined a number of recently published enhancements to the BM25 ranking function [2]. Trotman et al. showed that the way the parameters of ranking functions were tuned had an impact on retrieval effectiveness. When each variant of BM25 was tuned optimally for a particular collection, comparisons between the different functions were more complex and conclusions were harder to draw from results.

2.2. Query Expansion

Two commonly used approaches for query expansion are relevance model [3] and entity linking [4].

In the language modelling framework, the query representations can be formally augmented using a relevance modelling approach [3]. This approach has been put forward as a strong benchmark in past information retrieval research investigating the effectiveness of query expansion approaches [5]. The idea for relevance feedback is to take results that are initially returned from a given query, to use information about whether those results are relevant to perform a new query. It is often reasonable to assume that the top k documents are relevant and the bottom documents less so. In this case the user need not manually identify relevant and irrelevant documents as they are implicit in. This approach to feedback is known as pseudo-relevance feedback. Pseudo-relevance feedback approach uses retrieved documents to estimate the query topic. Top k documents are used for determining the expansion terms and are used in combination with the original query (the RM3 variant).

Several techniques were recently proposed for annotating documents with entities. These techniques usually rely on features extracted from Wikipedia for performing the disambiguation. To find entities in the text we have used TagMe [6], a state-of-the-art entity linking system based on a knowledge graph extracted from Wikipedia, whose main benefits are its performance on short texts and its speed. Wikipedia inlinks are explored for detecting and linking the entities present in search queries. The mentions are detected based on the probability that the mention appears as an anchor link on Wikipedia.

2.3. Clustering

Although we briefly considered a hierarchical clustering method to map out passages to page skeletons, which are both hierarchical due to their recursive structure, we decided to keep clustering non-hierarchical by attempting to cluster paragraphs to different section names. Though the section names can be hierarchical we utilized the flat representation of the TREC CAR pages to represent section names as strings.

While we used k-means with tf-idf similarity, a statistical model might also be able to function similarly. For example, Professor Douglas Baker created a naive Bayes model for clustering of words into word classes. It may be more accurate to call the method classification as it utilizes supervised training example to determine the statistical model and the likelihood of a word "belonging" in a certain group [15]. Our model uses tf-idf vectorization which is somewhat similar; instead of using training data to get an understanding of "belonging", tf-idf is based on the rarity of words in the corpus or other training data, but it would be possible to use the word classes of this method in lieu of tf-idf.

DBSCAN, like k-means, is a distance-based clustering method and widely used. The number of clusters is decided based on properties of the data, specifically with density of text. Unlike kmeans this model is soft clustering, in that documents can be decided to not belong in any particular cluster [16]. With our implementation, we know how many clusters should exist in the data, and we are expecting the input data to generally consist of passages assigned to the right page name if not necessarily the right section name, so these features of the method are not useful.

3. APPROACH

3.1. Okapi BM25

Okapi BM25 [10, 11], a probabilistic model of information retrieval used as a baseline approach. This method has shown very considerable benefits, enabling the development of effective weighting functions based on the three variables considered (term frequency within documents and queries and document length).

$$\mathrm{rsv}_q = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\mathrm{avgdl}}\right)}$$

where f(qi,D) is qi's term frequency in the document D, $| D |$ is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. k1 (1.2), b (0.75) are constants. IDF (qi) is the inverse document frequency weight of the query term.

3.2. BM25+

BM25+ [1,8] is an extension of BM25. BM25+ [8] addresses one pitfall of the standard BM25 [10,11] in which the component of term frequency normalization by document length is not properly lower-bounded; because of this drawback, long documents which do match the query term can often be scored unfairly by BM25. as having a similar relevancy to shorter documents that do not contain the query term at all. The scoring formula of BM25+ [1,8] only has one additional free parameter δ (a default value is 1.0 in absence of a training data. However, the best value for δ is 0.5 in our experiments) as compared with BM25:

$$\mathrm{rsv}_q = \sum_{i=1}^{n} \mathrm{IDF}(q_i) \cdot \left(\frac{f(q_i,D)\cdot(k_1+1)}{f(q_i,D)+k_1\cdot\left(1-b+b\cdot\frac{|D|}{\mathrm{avgdl}}\right)} + \delta \right)$$

Experiments on TREC data show that this function gives similar results as BM25 because of the shorter paragraph length.

3.3. LM-DS

Dirichlet smoothing is a standard method of interpolating between the collection model and the document model..The Dirichlet prior retrieval method is one of the best performing language modeling approaches [8]. This method uses the Dirichlet prior smoothing method to smooth a document language model and then ranks documents per the likelihood of the query per the estimated language model of each document. With a notation, consistent with those in the pivoted normalization and Okapi formulas, Petri et al. [18] give the derivation as:

$$rsv_q = L_q \, . \log\left(\frac{\mu}{L_d + \mu}\right) + \sum_{t \in q} tf_{tq} . \log\left(\frac{tf_{td}}{\mu} . \frac{L_c}{cf_t} + 1\right)$$

Where L_q is the length of the query, μ is a tuning parameter, tf_{tq} is the number of times the term occurs in the query, L_c is the length of the collection (in terms), and cf_t is the number of times the term occurs in the collection (the collection frequency). Experiments on TREC data show that this function, LM-DS, outperforming BM25.

3.4. Improved TF-IDF $TF_{1\cdot\delta\cdot p} \times IDF$

Rousseau and Vazirgiannis [14] suggest that nonlinear gain from observing an additional occurrence of a term in a document should be modeled using a log function. Following BM25+, they add μ to ensure there is a sufficient gap between the 0^{th} and 1^{st} term occurrence. Using these rules of combination and assuming log is the natural log, a combined formula for $TF_{1\cdot\delta\cdot p} \times IDF$ is generated as:

$$rsv_q = \sum_{t \in q} \ln\frac{N+1}{df_t} . \left(1 + \ln\left(1 + \ln\left(\frac{tf_{td}}{1 - b + b \, . \left(\frac{L_d}{L_{avg}}\right)} + \delta\right)\right)\right)$$

Experiments with the $TF_{1\cdot\delta\cdot p} \times IDF$ and other combinations of functions conducted on the TREC data suggest that this method is the best performing method among all BM25 improvements.

3.5. Entity Linking using TagMe

TagMe [6] is an annotator that links words to Wikipedia pages (i.e., to Wikipedia page URLs). TagMe works, broadly speaking, by jointly maximizing the coherence of all the (generic) entities e1, e2… ek spotted in a sentence S. By coherence is understood a measure of topical relatedness among such terms and Wikipedia entries. TagMe is available as a (RESTful) web service. The abstract and categories parameters are used for expansion terms. The abstract parameter is used, for each disambiguated spot, to include the abstract of the related Wikipedia page. The words in the abstract are then used in text expansion. The categories parameter includes the list of categories which the related Wikipedia page belongs to. The list of categories is provided by DBpedia.

3.6. Query Expansion with Relevance Model

Pseudo-relevance feedback (PRF) approach uses retrieved documents to estimate the query topic. It is a well-known method for improving retrieval effectiveness [19]. It is mainly used to solve the issue that user's query is always too short to describe the real intent clearly. Traditional PRF algorithms are based on the assumption that the top-ranked documents from an initial search are relevant to the query [20]. The high frequency words in these documents can expand the original query and retrieve again. However, the assumption is always invalid, and the performance of the PRF will be reduced. We are using top-100 and top-1000 documents to determine the expanded query terms and using those in combination with the original query (the RM3 variant). The techniques that are used to improve the relevance feedback process include number of top-ranked documents, iterations, term weighting, phrase versus single term and query expansion using entity linking.

3.7. Clustering

We implemented k-means clustering as a method to re-rank and reassign passages to section names at the page level. It is designed to work at the end of the pipeline, accepting input from the previous state of the pipeline either after one of the implemented ranking metrics or a complete .run file. Like the previous methods the clustering also uses the outline and paragraph .cbor files provided for this TREC task, though page information is loaded as well as section name information from the outline file. Two final input parameters are integer values for the number of extra clusters for k-means (the number of clusters for k-means on a particular page is the number of section names in a page plus this value) and a cutoff point (integer) for the number of passages extracted in the rankings generated from the previous methods. For example, for a cutoff point of 10, only the top 10 paragraphs (according to the previous methods) from each ranking are taken from each section name in a page to use in the clustering step. This is important to keep the k-means problem fast and meaningful; high values both increase the number of incorrect paragraphs being used to cluster and possibly increase the dimensionality of the distance metric to ineffective levels.

With inputs and data squared away, what remains is using tf-idf vectorization as a euclidean distance metric using the methods available in the scikit-learn (sklearn) library for Python 3. A list of strings, specifically each paragraph's text and the section name, are vectorized using cosine similarity in sklearn's Tfidfvectorizer. The number of clusters is as described above, a maximum of 300 iterations is used per k-means instance, and for each page the clustering algorithm is run five times to reduce the chance of converging to sub-optimal local minima based on the starting points. To assign section names to clusters pairwise cosine similarity is calculated between the section name and clusters of documents combined together in a bag of words format. Finally, a ranking of passages assigned to a section name is made with simple cosine similarity and the results are written in TREC CAR run file format.

4. EXPERIMENTAL EVALUATION

4.1. Dataset

Our experiments are conducted on TREC CAR Track datasets. TREC CAR Tracks use a variety of web corpora: Corpus-v1.4, Release-v1.4, Halfwiki-v1.4 and Test200-v1.4. Corpus-v1.4 provide the paragraph collection covering both test and train articles. Release-v1.4 contains 50% of Wikipedia articles matching selection criteria and is dedicated for training purpose. Halfwiki-v1.4 is minimal processed version of 50% of Wikipedia for training. Test200-v1.4 contain manual selection of 200 pages with outlines and qrels file for test and paragraph file provided for training. All our experiments are based on Test200-v1.4 for the input outlines and qrels and Corpus-v1.4 for passages. There are 2382 queries in Test200 and 7 million passages in Corpus-v1.4.

4.2. Evaluation

We have evaluated the results using different retrieval functions with and without expansion models and re-rank functions. We have accessed our system in terms of different evaluation measures, such as MAP, precision@k, R-precision, mean reciprocal rank (mrr) and so on. However, we have used map, p@5 and r-precision to depict the results of our experiments.

The mean average precision for a set of queries is computed as the mean of average precision scores for each query

$$MAP = \frac{\sum_{q=1}^{Q} \mathrm{AveP(q)}}{Q}$$

where Q is the number of queries.

R-precision requires knowing all documents that are relevant to a query. The number of relevant documents R, is used as the cutoff for calculation, and this varies from query to query. Empirically, this measure is often highly correlated to mean average precision.

Precision at k documents (P@k) is an another useful metric which corresponds to the number of relevant results on the first search results page.

4.3. Results

We use MAP, P@5 and R-PREC as the evaluation measures. The optimal performance for some of the pipelines is summarized in table 1. Of all the results for the test 200 queries and 7 million corpus-v1.4 passages, the performance of $TF_{1 \cdot \delta \cdot p} \times IDF$ is significantly better than that of LM-DS and BM25 Lucene implementations. The result from Dirichlet using TagMe expansion is almost as same as Dirichlet without expansion. Our analysis on the similar results show that the expanded words on one set of queries are useful while on the other set are not so useful and these two balance out to give us results similar to LM-DS without expansion.

Table 1. EVALUATION RESULTS FROM TESTING ON TEST 200 AND CORPUS-V1.4

Method	MAP	P@5	R-PREC
Lucene BM25	0.126	0.062	0.099
LM-DS	0.157	0.098	0.137
Improved TF-IDF	0.165	0.102	0.129
LM-DS with Query Expansion	0.154	0.092	0.132

Table 2 depicts the Mean Average Precision paired t-tests. Method 1 is Baseline BM25, method 2 is $TF_{1 \cdot \delta \cdot p} \times IDF$ and method 3 is clustering using $TF_{1 \cdot \delta \cdot p} \times IDF$f. Tests whether there is a statistically significant difference in the mean average precision output of method 2 compared to method. 1. A p-value of 0.05 or less indicates there is a significant (95% or greater confidence interval). There is not a significant enough difference between the two information retrieval methods, but the k-means re-ranking method has a significantly worse mean average precision.

Table 2. PAIRED T-TESTS (MAP) RESULTS

	Paired t-tests (Mean Average Precision)		
Method 1 (Baseline)	**Method 2**	**P-value**	**Significant?**
BM25 Ranking	Tf-idf ranking	0.7005156138	No
BM25 Ranking	Tf-idf k-means reranking	4.97E-119	Yes
Tf-idf ranking	Tf-idf k-means reranking	4.37E-119	Yes

Table 3 depicts R-precision paired t-tests. Tests whether there is a statistically significant difference in the R-precision (precision at a cutoff R where R = the number of relevant documents) compared to method 1. A p-value of 0.05 (95% confidence interval) indicates there is a significant difference. There is not a significant enough difference between the two information retrieval methods, but the k-means re-ranking method has a significantly worse mean average precision. The P-value for R-precision is several orders of magnitude larger than in table 2 but the P-value is so small it does not make much of a difference.

Table 3. PAIRED T-TESTS (RPREC) RESULTS

	Paired t-tests (Rprec)		
Method 1 (Baseline)	**Method 2**	**P-value**	**Significant?**
BM25 Ranking	Tf-idf ranking	0.6719616802	No
BM25 Ranking	Tf-idf k-means reranking	4.69E-78	Yes
Tf-idf ranking	Tf-idf k-means reranking	9.57E-78	Yes

Fig 1 and 2 show that the MAP and R-PRECISION reveals no significant differences between our best retrieval method and the BM25 benchmark.

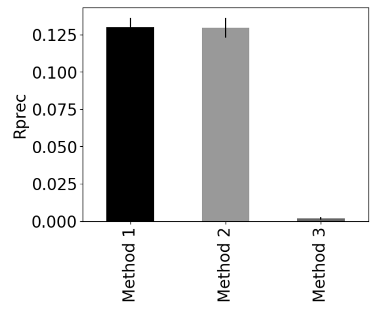

Figure 1. Mean retrieval effectiveness (MAP) with standard error bars

.

Figure 3 shows the difficulty success percentiles of method 1 compared to the others for mean average precision. Method 3 is mostly only succeeding at queries that method 2 handles best, the effectiveness of Method 2 drops slightly in the 75-95 percentile range. Difficulty success columns with method 2 as the primary method (not shown) confirmed that method 1 does outperform method 2 in this range even when the difficulties are ordered according to the difficulties as defined by method 2's evaluation of difficulty.

Figure 4 shows the difficulty success percentiles of method 1 compared to the others for R-precision. Method 3 seems to outperform the other methods on R-precision for more difficult measures (see the 5-25% and 50-75% percentiles). "This is likely just be due the cutoff of documents that K-means reranking looks at (set to 10 in this example) essentially means the precision for k-means will always be precision@10. Therefore, if the R-prec metric in the evaluation framework doesn't penalize methods for quitting at 10 instead of reaching the number of relevant documents retrieved, this would result in k-means reranking having artificially higher success just due to the nature of the precision-recall curve."

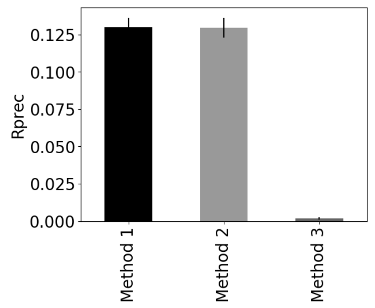

Figure 2. Mean retrieval effectiveness (R-PRECISION) with standard error bars

.

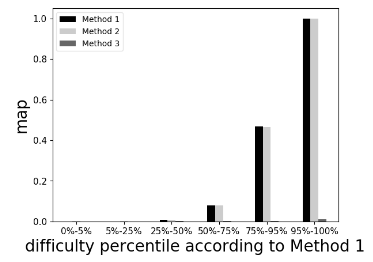

Figure 3. Difficulty percentile according to Method 1

.

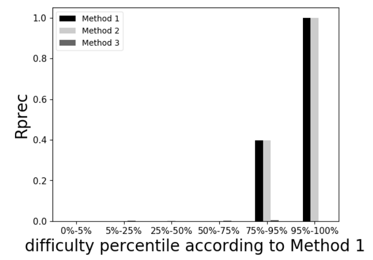

Figure 4. Difficulty percentile according to Method 1

.

5. CONCLUSION

In this paper, we propose different approaches to solve the TREC-CAR track based upon different retrieval methods, entity linking and machine learning techniques. We demonstrate how concepts from entity linking and machine learning can be incorporated within different IR frameworks.

In different collections(test200 and benchmarkY1), the improved tf-idf was the stronger performer compared to other variants of BM25. A particular retrieval method when used with or without entity-linking give almost the same results.

One limitation of query expansion using entity-linking is that it depends on the accuracy and success of the entity annotations and mentions linking. The results are also affected by the entities that were detected but are not in the paragraph collection. An interesting line of work could involve incorporating session-level features into the entity linking process in order to make use of a richer source of contextual information. Using additional entity properties is another interesting direction for further research.

With another month or two to work, learning how to use the threading library in the python standard library would be useful. There are at least four possible tasks that could work with multiple threads: annotation with tagme, our page-specific k-means clustering implementation, and two subtasks in the setup of the ranked retrieval methods, specifically text processing of paragraph text and section names with the porter stemming algorithm and calculating scores of the similarity between section names and passages with the ranking method(s) being used. There were two significant flaws in the implementation of k-means clustering using cosine similarity. First, the clustering algorithm used only the words in the current page's passages to learn the cosine similarity of words. This might or might not work well in an environment where the documents in the page all belonged to the page. This might sound like equivocation, but that's part of the problem; there's little we can do to effectively evaluate the metric if we train on new data at each step, save for compare the results to the qrel results page by page. We could train a k-means classifier over one of the provided sets of training data, in particular the largest data once and creating a cache file for it, but there is a better option. Using data output from Google's word2vec [17] was suggested as a better alternative to using tf-idf weighting. Google's word2vec is probably a better tool for all of what we sought to accomplish with k-means clustering. It's engineered to efficiently learn a large neural network of words on large training datasets (such as the training half-wiki file provided for training use in TREC CAR), supports parallelization for faster training, and outputs a file that serves as a vector representation of words and can output cosine similarity between any two words. There's even an option to set a cap on the dimensionality of word vectors, which would allow us to experiment with increasing dimensionality (and with it, likely better results) until we reach the point where the curse of dimensionality starts to hinder the effectiveness of our method.

6. CONTRIBUTIONS

Gaurav implemented different retrieval functions, framework for testing approaches, indexing, text processing algorithms and multi threaded implementation to process 700,000 documents per core. Shilpa implemented entity-linking query expansion using TagMe, relevance feedback model and Interpretation finding algorithm. Ran tests for benchmarkY1 train and test data. Colin wrote about clustering (and the implemented k-means clustering method) in related work, approach, future work/conclusion, and part of evaluation (t-tests, std error columns, difficulty columns, and generating all said results with minir-plots). Assisted Shilpa with proofreading and LaTeX formatting of paper.

References

[1] A. Trotman, A. Puurula, and B. Burgess. Improvements to BM25 and language models examined. In ADCS 2014, pages 58–65, 2014.

[2] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1995. Okapi at TREC-3. NIST SPECIAL PUBLICATION SP 109–109.

[3] V. Lavrenko and W. B. Croft. Relevance-Based Language Models. In Proceedings of the ACM SIGIR 01 conference, pages 120–127, 2001.

[4] Wei Shen, Jianyong Wang and Jiawei Han. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINNERING VOL: 27 NO: 2 YEAR 2015.

[5] Y. Lv and C. Zhai. Positional relevance model for pseudo-relevance feedback. In SIGIR '10, SIGIR '10, pages 579–586, New York, NY, USA, 2010. ACM.

[6] Ferragina, P. and Scaiella, U., 2010, October. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 1625-1628). ACM.

[7] Hasibi, Faegheh, Krisztian Balog, and Svein Erik Bratsberg. "Entity linking in queries: Tasks and evaluation." In Proceedings of the 2015 International Conference on The Theory of Information Retrieval, pp. 171-180. ACM, 2015.

[8] H. Fang, T. Tao, C.X Zhai. A formal study of information retrieval heuristics. Proceedings of the 27th International Conference on Research in Information Retrieval (SIGIR '04), pp. 49–56, 2004.

[9] Brandão, Wladmir C., Rodrygo LT Santos, Nivio Ziviani, Edleno S. Moura, and Altigran S. Silva. "Learning to expand queries using entities." Journal of the Association for Information Science and Technology 65, no. 9 (2014): 1870-1883.

[10] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In Proceedings of SIGIR'94, pages 232–241, 1994.

[11] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In TREC '94, pages 109–126, 1994.

[12] Y. Lv and C. Zhai. Lower-bounding term frequency normalization. In CIKM '11, pages 7–16, 2011.

[13] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In Proceedings of SIGIR'01, pages 334–342, Sept 2001.

[14] F. Rousseau and M. Vazirgiannis. Composition of TF Normalizations: New Insights on Scoring Functions for Ad Hoc IR. In Proc. of SIGIR, 2013.

[15] Baker, L. Douglas, and Andrew Kachites McCallum. "Distributional clustering of words for text classification." In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, pp. 96-103. ACM, 1998.

[16] Ester, Martin et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In KDD'96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp 226-231. AAAi press 1996.

[17] "word2vec: tool for computing continuous distributed representatives of words." Apache License 2.0 https://code.google.com/archive/p/word2vec/

[18] Petri, M., J.S. Culpepper, A. Moffat, Exploring the magic of WAND, ADCS 2013, p. 58-65

[19] Y. Li, R. W. P. Luk, E. K. S. Ho, and F. L. Chung. 2007. Improving Weak Ad-Hoc Queries using Wikipedia as External Corpus. In Proceedings of SIGIR.

[20] Yang Xu, Gareth J. F. Jones, and Bin Wang. 2009. Query Dependent Pseudo-Relevance Feedback based on Wikipedia. In Proceedings of SIGIR.