Benchmark for Complex Answer Retrieval

Federico Nanni
University of Mannheim
federico@informatik.uni-mannheim.de

Bhaskar Mitra
Microsoft and University College London
bmitra@microsoft.com

Matt Magnusson
University of New Hampshire
magnusson3@gmail.com

Laura Dietz
University of New Hampshire
dietz@cs.unh.edu

ABSTRACT

Providing answers to complex information needs is a challenging task. The new TREC Complex Answer Retrieval (TREC CAR) track introduces a large-scale dataset where paragraphs are to be retrieved in response to outlines of Wikipedia articles representing complex information needs. We present early results from a variety of approaches – from standard information retrieval methods (e.g., TF-IDF) to complex systems that adopt query expansion, knowledge bases and deep neural networks. The goal is to offer an overview of some promising approaches to tackle this problem.

1 INTRODUCTION

Over the last two decades, research in information retrieval (IR) has developed a variety of approaches for answering queries regarding precise facts – such as "Population New York City", "Who is Bill de Blasio?" or "Neighborhoods in Manhattan" – via the identification, extraction, and synthesis of pieces of information from textual sources. However, for more complex queries, such as "Benefits of immigration for NYC culture" current systems still rely on presenting the traditional ten blue links to the user as an answer.

To solicit works in this direction, and following the output of the recent SWIRL 2012 workshop on frontiers, challenges, and opportunities for information retrieval report [2], a new TREC track on Complex Answer Retrieval[1] (TREC CAR) for open-domain queries has been recently introduced [10].

The task and related dataset are based on the assumption that each Wikipedia page represents a complex topic, with further details under each sections. Accordingly, paragraphs contained in a section such as "Cultural diversity – Demographic" of the page "Culture of New York City", offer one aspect of the open-domain query "Culture of New York City". The goal of the task is presented as such: given an outline of a page (in the form of the page title and hierarchical section headings), retrieve a ranking of passages for each section. While assessed manually in the future, in this work, a passage is relevant for a section if and only if it is contained in the original article in the corresponding section.

Contribution. Many different approaches can be applied to this problem. This can address the ways a query can be expanded (using textual or structural information from knowledge bases), the way query and passages can be represented as vectors (e.g., word embedding vectors), and applications of deep neural networks and learning to rank. Given the recent release of the dataset, with this work we intend to support the future participants of the TREC CAR task by studying the performance of a variety of methods—to highlight which of them may be the most promising directions.

Additionally with the publication of this paper, we make our code and data available online.[2]

Outline. We give an overview of related work in Section 2, describe the data set and our experimentation environment in Section 3 and 4. Section 5 provides details of the approaches. We evaluate empirically in Section 6 before concluding the paper.

2 RELATED WORK

A wide variety of approaches are applicable to the TREC CAR problem. In the following, we cover three central ones, namely passage retrieval, query expansion using knowledge bases and the recent advancement in the use of deep neural network models for information retrieval.

Passage Retrieval. Passage retrieval is often cast as a variation on document retrieval, where the document retrieval model is applied only to a fragment of the text. The applications include search snippet detection, which aims to summarize the query-relevant parts of a document. Scores under the passage model can be combined with those from the containing document to improve performance [6] or to include quality indicators [5]. These approaches have been adapted to retrieve answers for questions [1]. Passage retrieval models can be extended to combine terms and entity-centric knowledge [8, 11]. For certain Wikipedia categories, template of articles can be extracted and automatically populated [24]. Furthermore, Banerjee and Mitra [4] found that training a lexical classifier per section heading obtains good results for article construction.

Neural-IR. With the comeback of neural networks, the IR community is exploring pre-trained word-vector approaches as well as dedicated neural networks for ranking. Pre-trained word- and entity-embeddings are publicly available in the form of word2vec,[3] GloVe,[4] DESM,[5] NTLM,[6] wiki2vec,[7] and RDF2Vec[8] vectors. Much

[1] http://trec-car.cs.unh.edu/

[2] https://federiconanni.com/trec-car-benchmark/
[3] https://code.google.com/archive/p/word2vec/
[4] http://nlp.stanford.edu/projects/glove/
[5] https://www.microsoft.com/en-us/download/details.aspx?id=52597
[6] http://www.zuccon.net/ntlm.html
[7] https://github.com/idio/wiki2vec
[8] http://data.dws.informatik.uni-mannheim.de/rdf2vec/

ICTIR '17, October 1-4, 2017, Amsterdam, The Netherlands

978-1-4503-4490-6/17/10…$15.00
DOI: https://doi.org/10.1145/3121050.3121099

of the work in this area focuses on the applications of these shallow distributional models to IR tasks, although recently deeper architectures have also been investigated [18, 26].

Query Expansion with KB. Pseudo relevance feedback [15] is one of the most popular query expansion methods in which frequent words in top documents of a first retrieval run are extracted. This idea is generalized to expansion with multiple sources [7] based on terms and phrases. Recent developments in entity linking algorithms and object retrieval make it feasible to tap into the rich information provided by KBs [9, 14, 17], and exploit disambiguation and confidences for query and document representation [13, 21]. Further work on entity aspects [22] and effective learning-to-rank approaches for latent entities [25] are a promising avenue.

3 DATA SET

To study complex answer retrieval, the TREC CAR organizers extract a large and comprehensive benchmark from Wikipedia articles [10].[9] English Wikipedia is processed to separate articles into outlines of hierarchical section headings and contained paragraphs (discarding info boxes, images and wrappers - v1.4). The hierarchical outlines are provided as complex information needs, where for each heading a ranking of paragraphs is to be retrieved. Automatic relevance data (qrels files) are provided based on whether a paragraph was listed in a given section.

Train data. For 50% of all articles, both outlines and articles are made available as training data for supervised machine learning as well as a resource for the Rocchio method discussed below.

Test200. As the track focuses on knowledge-centric topics (in contrast to biographies), 200 manually selected outlines are provided as a representative test collection. Together, these 200 outlines include approximately 2300 headings, each resulting in a query.

4 EXPERIMENTATION ENVIRONMENT

We focus on the task of retrieving and ranking paragraphs for each heading of the outline. We consider the 2300 sections in Test200 corpus: First we experiment in a "safe" experimentation environment before applying approaches to the entire collection of seven million paragraphs. The goal is to simulate a noisy candidate generation method, which is guaranteed to include all relevant paragraphs for the heading with a set of nearly-relevant negatives.

For each heading, we construct a **train set** by selecting, for every true paragraph under the heading, five paragraphs from different sections of the article, and five paragraphs from a different article.

Similarly we construct a **test set** that includes all paragraphs from the article, as well as the same amount of paragraphs drawn from other articles. All articles are provided in random order. On average this process yields a mean of 35 paragraphs per section.

5 EXAMINED APPROACHES

The setup of TREC CAR is a bit unusual as all headings of an article are given at once, with the goal of producing a separate ranking for each heading. In this early work, we are breaking each outline into several independent queries, one per heading h as follows: We identify the path from the heading to the root, and concatenate all all headings together with the page title to obtain the query. For example, if h corresponds to heading H2.3.4 the query is the concatenation of H2.3.4, H2.3, H2 and the page title.

Based on these queries, we experiment with different query expansion approaches and vector space representations of queries and paragraphs (TF-IDF, GloVe embeddings and RDF2Vec embeddings). We examine BM25, cosine similarity, learning to rank [16] and a state-of-the-art neural network model [19].

5.1 Query Expansion Techniques

We experiment three different query expansion approaches. We combine them with other scores, to, for instance, obtain RM3.

Expansion terms (RM1). Feedback terms are derived using pseudo relevance feedback and the relevance model [15]. Here we use Galago's implementation[10] which is based on a Dirichlet smoothed language model for the feedback run. In the experimental setting, we achieve the best performance expanding the queries with top 10 terms extracted from the top 10 feedback paragraphs.

Expansion entities (ent-RM1). Another way of expanding queries is by retrieving relevant entities. As for retrieving supporting terms, we derive a set of feedback entities by a search of the index using the heading-query and deriving several entities. In the experimental setting, best performance are achieved using 10 entities.

Paragraph Rocchio. Inspired by the work of Banerjee and Mitra [4], we retrieve other paragraphs, which have an identical heading to our heading-query, from folds 1 to 4 of the collection (omitting the fold where test200 originates from). For example, given a query such as "Demographic", regarding the entity United States, we collect supporting paragraphs from the pages of other entities (e.g., United Kingdom), which have as well a a section titled "Demographic". Headings are pre-processed with tokenisation, stopword/digit removal and stemming. This way, we can retrieve at least one supporting paragraph for 1/3 of our heading-queries. In the experimental setting, we test expansion using from 1 to 100 supporting passages and we obtain best performance expanding the query with 5 passages.

5.2 Vector Space Representations

We study three variations for representing the content in the vector space model.

TF-IDF. Representing each word in the vocabulary as its own dimension in the vector space, queries and paragraphs are represented as their TF-IDF vector. We are using the logarithmic L2-normalised variant. We perform stemming as a pre-processing step.

Word Embeddings. Using the pre-trained word embedding GloVE [20] of 300 dimensions, every word w in query or paragraph is represented as a K-dimensional vector $\vec{w}$. A vector representation for the whole paragraph $\vec{d}$ (complete query $\vec{q}$) is obtained by a weighted element-wise average of word vectors $\vec{w}$ in the paragraph (query). To give more attention to infrequent word, we use the TF-IDF of each word w to weights.

$$\vec{d} = \frac{1}{|d|} \sum_{w \in d} \text{TF-IDF}(w) \cdot \vec{w}$$

Entity Embeddings. Queries and paragraphs are represented as their mentioned DBpedia entities, using the entity linker TagMe

[9] Dataset available at: http://trec-car.cs.unh.edu/datareleases/v1.4-release.html

[10] lemurproject.org/galago.php

[12] (with default parameters). Next, we obtain latent vector representations $\vec{e}$ of each linked entity e using pre-computed **RDF2Vec** 300d entity embeddings [23]. Vector representations of paragraphs $\vec{d}$ (queries $\vec{q}$) are computed by a weighted element-wise average of entity vectors $\vec{e}$. By casting a paragraph as a bag-of-links we adapt TF-IDF to entity links (link statistics from DBpedia 2015-04 [3]:

$$\vec{d} = \frac{1}{|\{e \in d\}|} \sum_{e \in d} \text{TF-IDF}(e) \cdot \vec{e}$$

5.3 Ranking Approaches

We include four different ranking approaches.

Okapi BM25. Results are ranked using Okapi BM25 with k_1=1.2 and b=0.75, using the implementation of Lucene 6.4.1. Porter stemming and stopword removal was applied to paragraphs and queries.

Cosine Similarity. Paragraphs are ranked by cosine similarity (**cs**) between vector representations of the query and the paragraph.

Learning to Rank. We combine the ranking scores of different baselines with supervised machine learning in a learning-to-rank setting, for producing a final ranking of relevant paragraphs. We use RankLib [11] with 5-fold cross validation using a linear model optimized for MAP, trained with coordinate ascent.

Deep Neural Network. The Duet model is a state-of-the-art deep neural network (DNN) recently proposed by Mitra et al. [19] for ad-hoc retrieval. The Duet architecture learns to model query-paragraph relevance by jointly learning good representations of the query and the paragraph text for matching, as well as by learning to identify good patterns of exact matches of the query terms in the paragraph text. We use the Duet implementation available publicly[12] under the MIT license for our experiments. Training on folds 1 to 4 of the collection, we only consider the first ten words for the query and the first 100 words for the passage as inputs. We use 64 hidden units in the different layers of the network, as opposed to 300 in the original paper, to reduce the total number of learnable parameters of the model. We trained the model for 32 epochs with a learning rate of 0.001 which was picked based on a subset of the training data. Each epoch was trained over 1024 minibatches, and each minibatch contained 1024 samples. Each training sample was a triplet consisting of a query, a positive passage, and a negative passage. The training time was limited to 60 hours.

6 EVALUATION

We present experiments both on a small set and on the full data set.

6.1 Experimentation Environment

The experimentation environment (Section 4) provides a "safe environment" by simulating a noisy candidate method for each section. Results are presented in Table 1. The approach *bm25 query only* sets the baseline of our work.

Not all query expansion approaches and vector space representation methods improve over this baseline. This is particularly true for query expansion with terms or entities (through RM3 = query + RM1) as well as RDF2Vec embeddings. On the contrary, the most promising results among the methods which employ cosine similarity as a ranking function, are obtained when the query is expanded with Rocchio vectors trained on paragraphs from sections with the same heading. This finding reconfirms the results of previous work on the automatic generation of Wikipedia articles based its structural information [4]. The results show that common traits between Wikipedia sections with the same heading are better captured using the TF-IDF word vector than through word- and entity-embedding vectors suggest that a possible improvement over these baselines could by obtained by training embeddings for this task.

In comparison to these unsupervised retrieval models, both supervised Learning to Rank and the Neural Duet model out-perform all previously described baselines. In particular, the Duet model yields a substantial improvement over all presented approaches, showing the potential of neural-IR for the task. It is important to remark that neural deep models take days to train even on a GPU. In addition they are data-hungry, with performances improving significantly with more training data as shown in Figure 1.

Table 1: Results on experimentation environment.

	MAP	R-Prec	MRR
BM25			
query only	0.304	0.225	0.388
TF-IDF (cs)			
query only	0.328	0.212	0.385
query + RM1	0.325	0.206	0.385
query + Rocchio	0.401	0.286	0.467
GloVe (cs)			
query only	0.329	0.210	0.387
query + RM1	0.255	0.148	0.305
query + Rocchio	0.350	0.236	0.410
RDF2Vec (cs)			
entity-query only	0.313	0.200	0.369
ent-query + ent-RM1	0.322	0.209	0.379
ent-query + ent-Rocchio	0.316	0.205	0.376
Learning to Rank			
all (cs) scores	0.412	0.295	0.478
Duet model			
query only	**0.465**	**0.359**	**0.552**

Table 2: Results of the initial candidate selection.

	MAP	R-Prec	MRR
BM25			
query only	**0.140**	**0.110**	**0.202**
TF-IDF (cs)			
query only	0.035	0.025	0.053
query + Rocchio	0.029	0.020	0.041

6.2 Experiments on Full TREC collection

Moreover, we conduct experiments on the entire paragraph collection.

Candidate Selection. Many of the previously presented methods, such as the Duet model, require a candidate generating method. We test three candidate methods: BM25, TF-IDF, and TF-IDF with Rocchio expansion. For each query, the methods produce a candidate

[11] lemurproject.org/ranklib.php

[12] https://github.com/bmitra-msft/NDRM/blob/master/notebooks/Duet.ipynb

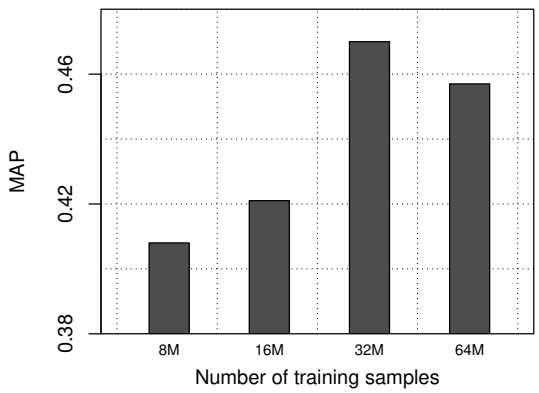

Figure 1: Effect of training data size on the performance of the Duet model. Training on four folds, reporting MAP on holdout fold.

Table 3: Results on the Paragraph Collection after candidate method. ▼ Worse according to paired-t-test with $\alpha = 5\%$.

	MAP	R-Prec	MRR	w/o BM25 MAP
BM25 candidate	0.140▼	0.110	0.202	-
theoretical upper-bound	0.382	0.382	0.537	0.382
TF-IDF (cs)				
BM25 + query + Rocchio	0.143▼	0.112	0.206	0.085
GloVe (cs)				
BM25 + query + Rocchio	0.150▼	0.119	0.217	0.082
Duet model				
BM25 + query only	**0.160**	**0.130**	**0.229**	0.094

set of 100 paragraphs. The results are presented in Table 2. While this is a challenging task, encouraging performance are obtained by *BM25 query only*, which we use in the following. Since not all relevant paragraphs are contained in the candidate set, the theoretically achievable performance of following methods is upper-bound by MRR is 0.537 and MAP/R-Prec is 0.382.

We evaluate the three best systems from Table 1 on the candidate set, combining candidate method BM25 and other components with learning to rank (5-fold cross validation) in Table 3. The combination of BM25 score and duet model is significantly outperforming all other methods, demonstrating the strength of neural method (although the Duet is by far the most expensive method to train). However, if the combination with the BM25 score is left out, all methods are significantly loosing in performance (see last column in Table 3).

7 CONCLUSIONS

In this paper, we present the performance of a variety of approaches, from established baselines to more advanced systems, in the context of the new TREC-CAR track on Complex Answer Retrieval.

Our results show that Neural retrieval methods provides best results only (!) when combined with the score of the candidate method. Among fast to train methods, we find that BM25 is a strong baseline, and that a Rocchio classifier based on headings is better than query expansion with pseudo-relevance feedback (RM3). We offer this empirical analysis as a complement to the publicly available TREC CAR dataset, to support future participants of the track and the IR community.

Acknowledgements

The publication is funded in part through the scholarship of the Eliteprogramm for Postdocs of the Baden-Württemberg Stiftung (project "Knowledge Consolidation and Organization for Query-specific Wikipedia Construction").

REFERENCES

[1] Elif Aktolga, James Allan, and David A. Smith. 2011. Passage reranking for question answering using syntactic structures and answer types. In *Advances in Information Retrieval*. Springer.
[2] James Allan, Bruce Croft, Alistair Moffat, and Mark Sanderson. 2012. Frontiers, challenges, and opportunities for information retrieval: Report from SWIRL 2012 the second strategic workshop on information retrieval in Lorne. In *ACM SIGIR Forum*, Vol. 46. ACM, 2–32.
[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.
[4] Siddhartha Banerjee and Prasenjit Mitra. 2015. WikiKreator: Improving Wikipedia Stubs Automatically.. In *ACL (1)*. 867–877.
[5] Michael Bendersky, W Bruce Croft, and Yanlei Diao. 2011. Quality-biased ranking of web documents. In *WSDM*.
[6] Michael Bendersky and Oren Kurland. 2008. Utilizing passage-based language models for document retrieval. In *Advances in Information Retrieval*. Springer.
[7] Michael Bendersky, Donald Metzler, and W Bruce Croft. 2012. Effective query formulation with multiple information sources. In *WSDM*.
[8] Roi Blanco and Hugo Zaragoza. 2010. Finding support sentences for entities. In *SIGIR*.
[9] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity Query Feature Expansion Using Knowledge Base Links. In *SIGIR*.
[10] Laura Dietz and Ben Gamari. 2017. TREC CAR: A Data Set for Complex Answer Retrieval. Version 1.4. (2017). http://trec-car.cs.unh.edu
[11] Laura Dietz and Michael Schuhmacher. 2015. An interface sketch for queripidia: Query-driven knowledge portfolios from the web. In *Proc. Workshop on Exploiting Semantic Annotations in IR*. ACM, 43–46.
[12] Paolo Ferragina and Ugo Scaiella. 2010. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM*. ACM.
[13] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *ICTIR*. 209–218.
[14] Alexander Kotov and ChengXiang Zhai. 2012. Tapping into knowledge base for concept feedback: leveraging conceptnet to improve search results for difficult queries. In *WSDM*.
[15] Victor Lavrenko and W Bruce Croft. 2001. Relevance based language models. In *SIGIR*.
[16] Hang Li. 2014. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies* 7, 3 (2014).
[17] Xitong Liu and Hui Fang. 2015. Latent entity space: a novel retrieval approach for entity-bearing queries. *Information Retrieval Journal* 18, 6 (2015).
[18] Bhaskar Mitra and Nick Craswell. 2017. Neural Models for Information Retrieval. *arXiv preprint arXiv:1705.01509* (2017).
[19] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *www*.
[20] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14.
[21] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document Retrieval Using Entity-Based Language Models. In *SIGIR*.
[22] Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2015. Mining, ranking and recommending entity aspects. In *SIGIR*.
[23] Petar Ristoski and Heiko Paulheim. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *ISWC*. Springer.
[24] Christina Sauper and Regina Barzilay. 2009. Automatically generating Wikipedia articles: A structure-aware approach. In *IJCNLP*.
[25] Chenyan Xiong and Jamie Callan. 2015. Esdrank: Connecting query and documents through external semi-structured data. In *CIKM*.
[26] Ye Zhang, Md Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, and others. 2016. Neural Information Retrieval: A Literature Review. *arXiv preprint arXiv:1611.06792* (2016).