TREMA-UNH at TREC 2018: Complex Answer Retrieval and News Track

Sumanta Kashyapi, Shubham Chatterjee, Jordan Ramsdell, Laura Dietz
{sk1105, sc1242, jsc57}@wildcats.unh.edu, dietz@cs.unh.edu
TREMA lab, University of New Hampshire, U.S.A

Abstract

This notebook describes the submission of team TREMA-UNH to the TREC Complex Answer Retrieval track and the TREC News track in 2018. Our methods focus on passage retrieval, entity-aware passage retrieval, and entity retrieval.

1 Introduction

Users expectations of search engines continue grow with advancements of search engines and retrieval models. We explore "search engines of the future" that not only rank documents according to relevance, but also present all relevant information in a compact manner from which users are able to synthesize knowledge easily. To accomplish this task, we train retrieval models to have a better understanding relevance for natural language.

The Complex Answer Retrieval (CAR)[1] track at the Text Retrieval Conference (TREC)[1] aims to address this scenario. Current retrieval systems provide good solutions towards passage retrieval for simple fact and entity-centric information needs. In contrast, CAR is about answering more complex information needs with longer answers. The formal task statement includes the two following tasks:

CAR Passage Task: Given an article stub Q, retrieve for each of its sections H_i, a ranking of relevant passages P. The passage P is taken from a provided paragraph corpus.

CAR Entity Task: Given an article stub Q, retrieve for each of its sections H_i, a ranking of relevant entities E. The entity E is taken from a provided Wikipedia corpus. Additionally for each entity E, a support passage from the passage corpus is to be identified that explains why the entity is relevant for the heading H_i on the stub.

We further participate in the NEWS track entity ranking task.

[1] https://trec.nist.gov

NEWS Entity Task: Given a news article with title and content that is annotated with entity links to a set of entities $\mathcal{E} = \{E1, E2, ...En\}$, the task is to rank the the given entities by importance for the article (i.e., saliency).

2 Overarching Approach

For each page, we use the section headings from the page's outline as queries. The headings are usually very short. Hence, using retrieval models without query expansion (such as BM25) may not yield good results since headings have potentially little textual overlap with their passages. To alleviate this problem, we experiment with various entity- and word-based query expansion methods. We observe the performance of each of the methods individually. However, since any single unsupervised method may not provide best performance on its own, we also experiment with machine-learned combinations of methods using learning-to-rank.[2]

A knowledge graph (KG) is used to enhance a retrieval models's results with information gathered from a variety of sources. The KG provides access to knowledge about entities and is derived from external sources such as Wikipedia. Here we use a Wikipedia dump[3] provided by the TREC CAR organizers that does not include pages from test queries. Such knowledge graphs hold a wealth of information which can be leveraged to solve information retrieval problems.

In Section 3.1 we first describe which unsupervised retrieval and expansion models were used for the passage task, and in Section 3.2 for the entity task.

3 Low-level Retrieval Features

Each of our approaches are based on a variety of unsupervised retrieval models and document indexes that we describe in the following.

Indexes: We create the following indexes for use with our retrieval methods.

- A paragraph index out of the text in passages of the `paragraphCorpus`.
- A page index out of all visible text on Wikipedia pages in `allButBenchmark`.
- An entity index out of the first paragraph, anchor text, and category info of Wikipedia in `allButBenchmark`.

Query models: Given a stub with page title T and a tree-shaped outline of headings $H1$, $H1.1$, $H1.2$, $H1.2.1$, $H2$,... there are different ways to derive queries for retrieval with BM25, QL, etc for text of a particular heading, such as $H1.2$. The simplest approach is what we call *section path queries*, which

[2] `http://lemurproject.org/ranklib.php`

[3] `unprocessedAllButBenchmark.v2.1.tar.xz`, Wikipedia dump from December 2016

concatenates the page title, T, with all parent headings (in this example $H1$), along with the heading itself ($H1.2$), to derive the query. However, more options are possible such, as as:

- Section Path: Concatenation of page title, parent headings, and heading (Example: $T, H1, H1.2$).
- Title: only the page title (T)
- Leaf: only the heading itself ($H1.2$)
- Internal: concatenation of the parent heading(s) ($H1$)
- All: concatenation of page title and all headings in the stub ($T, H1, H1.1, H1.2, H1.2.1, H2, \dots$)
- Subtree: concatenation of all headings in the subtree rooted by the heading ($H1.2, H1.2.1$)

The section path model is very popular among the TREC CAR participants and usually works best as a standalone method. However, it has been argued that learning a weighted combination between title, internal, and leaf query models could give potentially even better performance [2].

Retrieval and expansion models: Given a query model to transform the stub into a keyword query and an index, and we use different retrieval models to obtain low-level rankings (which we will combine with different learning to rank methods in the following). In particular we use as retrieval models:

- BM25, as implemented in Lucene (using default parameters).
- Query likelihood with Dirichlet smoothing, as implemented in Lucene.

We use as expansion models

- None: No expansion. Just uses the initial ranking.
- RM3: RM Relevance model expansion [3] using 20 top paragraphs/pages to expand with 20 terms. This is an RM1-style ranking that is intended to be combined with the basic ranking model (to yield RM3).
- ECM: A variant of RM3 representing a document as a bag-of-entity-link-targets. Uses top 100 paragraphs/pages.
- ECM-psg: Like ECM expansion but expands the keyword query with top 100 expansion entities.

Feature	MAP	Rprec	recip_rank
sectionPath-bm25-none	0.1291	0.1031	0.2006
sectionPath-ql-none	0.1232	0.0939	0.1888
sectionPath-bm25-rm	0.1038	0.0767	0.1657
sectionPath-ql-rm	0.0744	0.0493	0.1180

Table 1: Paragraph feature results, Y1 Train tree qrels

Ecm uses the entity context model in a non-standard way, where the whole page is used instead of short passages: After retrieving a feedback run of paragraphs or Wikipedia pages with BM25, each page is represented as bag-of-entity-link-targets. Using the relevance model [3], we compute the distribution over expansion entities $P(E|Q)$. We discuss two variants, in **ecm-passage** we expand the query with these entities to retrieve a new ranking. In **ecm** we use the distribution directly to produce a ranking of entities.

$$score(E|Q) = \sum_{D \in \text{ranking}} p(D|Q)p(E|D)$$

The name ecm is in attribution to the entity context model [4], which uses passages with contained entity mentions for new expansion models. Where Dalton et al. [4] uses passages surrounding entity links in a feedback run, the ecm-paragraph index allows us to directly retrieve relevant passages with entity-centric information.

We provide all low-level runs for benchmarkY1train, benchmarkY1test and benchmarkY2test online.[4]

3.1 Low-level Paragraph Retrieval Features

We use the following pararaph ranking and expansion models to derive features for learning-to-rank for the passage task (decribed in the following sections).

- sectionPath-bm25-none: Using section path queries, BM25 retrieval model (no expansion)
- sectionPath-ql-none: Using section path queries, Query likelihood retrieval model (no expansion)
- sectionPath-bm25-rm: Using section path queries, BM25-based RM3
- sectionPath-ql-rm: Using section path queries, query likelihood-based RM3

The results of low-level retrieval methods are presented in Table 1.

[4] Available at `http://trec-car.cs.unh.edu/runs/TREMA-UNH`

3.2 Low-level Entity Retrieval Features

We use the following ranking and expansion models to derive features for learning-to-rank for the entity task (decribed in the following sections).

- sectionPath-bm25-ecm: Using section path queries, BM25 ranking with ECM ranking based on BM25 page retrieval.
- sectionPath-ql-ecm: Same as above, but based on Query likelihood.
- all-bm25-ecm: Constructing a query from title and all headings on the outline, ECM ranking based on BM25 page.
- all-ql-ecm: Same as above, but based on Query likelihood.

These were then combined using learning-to-rank and a combined ranking was obtained. This model is trained on benchmarkY1-train, using the tree-level ground truth for the passage task.

4 UNH-p-l2r

Interpreting the set of rank scores (using Section 3.1) for a particular paragraph as features, we learn how to optimally combine these features into a weighted combination. We train the model using the coordinate ascent algorithm of RankLib's learning to rank implementation, optimized for mean average precision.

We train combinations of:

- sectionPath-bm25-none: BM25 retrieval model
- sectionPath-ql-none: Query likelihood retrieval model (no expansion)
- sectionPath-bm25-rm: BM25-based relevance feedback
- sectionPath-ql-rm: Query likelihood-based relevance feedback

5 UNH-e-l2r

Using Lucene indexes for entity, page and paragraph (as described in Section 3). We create features for each index by combining query-level, retrieval-model, and expansion-model methods as follows:

We train combinations of (across all choices of entity, page and paragraph indexes):

- sectionPath-bm25-ecm: Section-path BM25 retrieval model with ecm entity ranking
- sectionPath-ql-ecm: Same as above but with Query likelihood instead of BM25

- all-bm25-ecm: Query from title and all headings using BM25 retrieval model with ecm entity ranking
- all-ql-ecm: Same as above but with Query likelihood instead of BM25

These were then combined using learning-to-rank and a combined ranking was obtained. This model is trained on benchmarkY1-train, using the tree-level ground truth for the entity task.

The rankings were annotated with the highest ranked paragraph from an additional paragraph ranking as described below.

6 UNH-p-SDM

We use Lucene to index the TREC CAR 2017 paragraph corpus after stemming and removing stopwords. The UNH-p-SDM method is inspired by the Sequential Dependence model [5] in that it combines unigrams, bigrams, and windowed-bigrams. However, where the original approach by Metzler et al. used language models (such as query likelihood) as features, here we use BM25 as a basis for scoring the relevance of documents given a query. We do so by indexing the unigrams, bigrams, and unordered windowed-bigrams (with a window size of eight words) as document fields using Lucene. We tokenize and stem documents using Lucene's English analyzer. We then consider three features that score document relevance with respect to these fields:

- Unigram: Unigrams in the query are used to retrieve and score passages via their unigram fields.
- Bigram: Bigrams in the query are used to retrieve and score passages via their bigram fields with BM25/QL.
- Windowed-bigram: Unordered windowed bigrams in the query are used to retrieve and score passages via their windowed bigram fields.

We train a weighted combination of these three scores that best predicts the relevance of a passage. We do so by using RankLib with coordinate ascent, optimized for best MAP performance.

7 Joint Entity-Passage Methods

In this paper, we explore features that can be used to jointly score entities and passages. We do so using the following approaches.

Passage retrieval using entity features. Let f_e be an entity relevance feature. For each passage, p, let E be the set of entities contained in p. Then a feature that scores the relevance of p given the relevance entities can be represented as $f_p(p) = \sum_{e \in E} f_e(e)$. We are free to combine this feature with other passage features to rank retrieved passages according to relevance.

Entity retrieval using passage features. Let f_p be a passage relevance feature. For each entity, e, let P be the set of passages (retrieved with UNH-p-SDM) that contain at least one instance of e. We may sum over the scores of passages that contain e to produce a score for e: $f_e(e) = \sum_{p \in P} f_p(p)$.

7.1 Description of Passage Features

We consider the following passage features for use in our joint entity-passage methods:

UNH-p-SDM. We directly use the scores of passages obtained by the UNH-p-SDM (see section 6)) passage retrieval method as a passage feature.

SDM. Passages are scored using a standard SDM model under query likelihood, in the spirit of Metzler et al. [5].

We convert these passage features into into entity features as described above.

7.2 Description of Entity Features

We consider the following entity features for use in our joint entity-passage methods:

Entity Link Frequency. We score an entity with respect to the relative frequency at which it was linked to by passages in a candidate set of passage. We do so by summing over the total number of times an entity was linked by candidate passages, and then normalizing by the total number of entity links among all candidate passages.

Fielded Queries. Because our entity index represents a collection of Wikipedia pages, we can consider page attributes as document fields. We store the following page attributes as unigrams in document fields: (enwiki) categories, inlinks, outlinks, section headers, page name disambiguations, and page name redirects. We score the relevance of entities with respect to the query by using the standard BM25 model. This feature can also be transformed into a passage feature.

Global Entity Context. Whenever a passage links to an entity in Wikipedia, we create a pseudo-document that contains the unigrams, bigrams, and windowed bigrams derived from that passage. We consider the collection of all such pseudo-documents belonging to an entity as that entity's "context" in which it is mentioned. This differs from query-specific context models [4], in that we use one global entity context model derived from the corpus. For each query, we construct a unigram, bigram, and windowed bigram query that is used to retrieve pseudo-documents from the global entity context index.

Pseudo-documents are scored with respect to the standard BM25 model. Each pseudo-document represents a particular context in which an entity is mentioned in a passage, and we let an entity's score be equal to the highest scored context in which it was mentioned. We obtain one such score for each of the retrieval methods (unigrams, bigrams, and windowed bigrams). We treat each score as a separate entity feature.

All these features are used either directly as an entity feature, or they are transformed to a passage feature by summing over the relative entity frequencies of entities contained within a passage.

7.3 UNH-e-SDM

For our entity retrieval task, we derive an entity ranking (the UNH-e-SDM) from the scores of candidate passages using the UNH-p-SDM passage feature as described in Section 7.1. We score each candidate entity by summing over the scores of candidate passages that are linked to this entity. We retrieve candidate entities by retrieving entities linked to the top 100 candidate passages ranked using the UNH-p-SDM passage retrieval method.

7.4 UNH-p-Mixed

The UNH-p-mixed method is a passage retrieval method that utilizes all of the passage features described in section 7.1 (UNH-p-SDM and SDM). This method also uses all of the entity features described in section 7.2 (entity link frequency, the six fielded query features, and the three entity context features), which are transformed into passage features. As previously described, we score passages using entity features by summing over the scores of entities contained within each passage.

The top 100 candidate passages are retrieved using the UNH-p-SDM passage retrieval method, and all entities linked to these passages are used as candidate entities. Candidate entities and passages are scored using the above features. Finally, we learn a weighted combination of these passage features and transformed features using learning to rank (where queries and query relevance is derived from the section path model described previously). We find that UNH-p-SDM feature receives the highest weight.

7.5 UNH-e-Mixed

The UNH-e-mixed method is an entity retrieval method that utilizes all of the entity features described in section 7.2. These include entity link frequency, the six fielded queries (enwiki categories, inlinks, outlinks, section headers, page name disambiguations, page name redirects), the three entity context features, and all of the passage features described in section 7.1 (UNH-p-SDM and SDM). The passage features are transformed into entity features such that the score of each entity is equal to the sum of the scores of passages that link to the entity, as mentioned in Section 7.

Our candidate passages and entities are retrieved using the approach described in Section 7.4. We again use learning to rank (and the section path query model) to learn a linear combination of entity features and transformed passage features.

8 UNH-e-graph

We consider a graph where entities are nodes and paragraphs form edges between all nodes that are contained in the paragraph. Degree centrality, PageRank, personalized PageRank, HITS, or SALSA could be applied to this graph to identify important nodes. However, unsupervised graph walk methods have no knowledge about which edges are relevant for the query, and therefore suffer from concept drift.

We explore a novel, unpublished method for "Learning to Walk", where nodes and edges are associated with feature vectors that quantify their relevance for the query—these are derived by unsupervised rank scores such as BM25 or relevance models. It is possible to derive an optimization algorithm to optimize weight parameters for node and edge features, so that the entity ranking produced by applying degree centrality to the graph obtains the best MAP performance.

Similar to traditional learning to rank approaches, features are derived from different unsupervised ranking functions. We use indexes described in Section 2. Rankings of entities provide a feature vector for nodes; rankings of paragraphs provide a feature vector for edges. The Learning-to-Walk algorithm is used to train weight parameters to optimize for MAP on entity rankings. The Learning-to-Walk is trained with mini-batched coordinate ascent using five restarts on benchmarkY1train.

In contrast to our other methods, this one is trained with a custom version of a tree-qrels, created as follows: For all queries in the benchmarkY1train and benchmarkY1test we obtain the Wikipedia page. These pages are entity linked with DBpedia Spotlight. Then all predicted entity links are compared to link targets that were manually included in the page by Wikipedia editors. Spotlight links are only retained if the Wikipedia editor had included a link to the same target (anywhere on the page).

This process is very similar to how the official automatic qrels were created by the track organizers, with the difference that official qrels are based on entity links created by Wikipedia editors, where we extend this set with our spotlight-based heuristic. The main concern is that due to Wikipedia's editorial policies, only the first mention of an entity of a page is annotated with a hyperlink to the entity. However, central entities may get mentioned several more times on the page, possibly in other sections. Without fixing this ground truth, the concern was that when relevant entities are included in the ranking, we don't want the qrels file to (falsely) indicate that the entity is non-relevant.

We train several variations of this method and select the best variation for submission as "UNH-e-graph". We found in post-mortem experiments that including ecm-expansion would lead to even better performance. However, we were concerned that this feature would not generalize well to non-Wikipedia collections and did not include it in the variant submitted.

9 Support Passage for Entity Rankings

The entity ranking is augmented with support passages that are supposed to explain to a user how this entity is relevant for the heading H_i.

We first compute an entity ranking without support passages, then extend it as follows. We use a paragraph ranking obtained using one of our paragraph ranking methods. For every entity in the ranking, we return the first paragraph in the ranking which contains this entity. We use the entity-passage pairs to produce a run file in the TREC CAR entity ranking task format. The score of a paragraph given a query and entity in this combined run file is the original score of the entity.

10 Evaluation on TREC Complex Answer Retrieval

We evaluate the performance of our passage and entity retrieval methods with the following qrels files (all from the TREC CAR v1.2 data set) [1]:

- Y1 Train: `benchmarkY1train`, automatic tree-qrels, using 5-fold CV.
- Y1 Test: `benchmarkY1test`, automatic tree-qrels, trained on Y1 Train.
- Y2 Test: `benchmarkY2test`, manual assessments, trained on Y1 Train, best methods selected on Y1 Test.

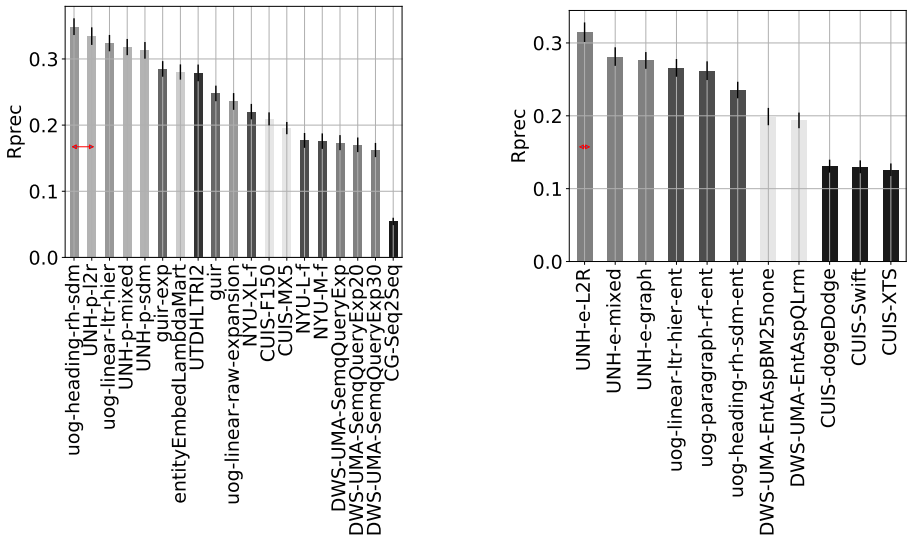
For each method, the top 100 paragraphs/entities were retrieved.

10.1 Results

We compare the performance of our passage retrieval methods using three benchmarks (Y1 Train, Y1 Test, and Y2 Test) in Table 3. Highest MAP values are depicted in bold. Results are presented in Table 2 and 3.

Among the submitted runs, the runs produced by UNH-p-l2r and UNH-e-l2r work best on Y2 Test. This is especially surprising for UNH-p-l2r for two reasons: (1) the method is fairly simple, including only BM25 and Query likelihood with and without RM3 expansion; (2) both were not necessarily performing best in terms of MAP or Rprec on automatic Y1 train and Y1 test benchmarks—in contrast to findings of the track organizers last year, that automatic and manual assessments lead to nearly the same system rankings.

On the automatic benchmarks Y1 Train and Y1 Test, UNH-p-mixed and UNH-e-mixed methods have the highest performance with respect to RPrec and MAP. Interestingly, their performance is the lowest among all our submitted methods with respect to the manual benchmark (Y2 Test). We speculate that the NDCG measure on the automatic benchmarks are better correlated with good performance on manual assessments of Y2 test.



(a) Passage Task

(b) Entity Task

Figure 1: TREC CAR 2018 results for all participated systems, using the manual benchmark and Rprec as evaluation measure.

Figures 1 present evaluation results across all participating systems with respect to RPREC. As we see, among those that participated, our methods are either best or second best.

Table 2: Performance of entity retrieval methods. Y1 Train and Y2 Test benchmarks are derived from an automatically generated ground truth. The Y2 Test benchmark contains the TREC 2018 manual assessments.

Method	**Y1 Train**			**Y1 Test**			**Y2 Test**		
	MAP	**RPREC**	**NDCG**	**MAP**	**RPREC**	**NDCG**	**MAP**	**RPREC**	**NDCG**
UNH-e-l2r	0.15	**0.17**	**0.42**	**0.17**	**0.18**	**0.44**	**0.31**	**0.31**	**0.51**
UNH-e-graph	0.14	0.15	0.34	0.14	0.16	0.34	0.27	0.28	0.48
UNH-e-sdm	0.16	0.17	0.38	0.17	0.18	0.40	0.26	0.28	0.46
UNH-e-mixed	**0.16**	**0.17**	0.38	**0.17**	**0.18**	0.40	0.27	0.28	0.44
bm25-ecm	0.11	0.12	0.32	0.10	0.12	0.32	0.18	0.19	0.42
ql-ecm	0.11	0.13	0.33	0.11	0.13	0.32	0.17	0.19	0.40

Table 3: Performance of passage retrieval methods (submitted and low-level sectionpath baselines). Y1 Train and Y1 Test benchmarks are derived from an automatically generated ground truth. The Y2 Test benchmark contains the TREC 2018 manual assessments.

Method	**Y1 Train**			**Y1 Test**			**Y2 Test**		
	MAP	**RPREC**	**NDCG**	**MAP**	**RPREC**	**NDCG**	**MAP**	**RPREC**	**NDCG**
UNH-p-l2r	0.12	0.09	0.20	0.13	0.11	**0.27**	**0.34**	**0.33**	**0.58**
UNH-p-SDM	0.15	0.12	0.25	0.14	**0.13**	0.24	0.30	0.31	0.47
UNH-p-mixed	**0.16**	**0.14**	**0.25**	**0.15**	**0.13**	0.25	0.29	0.32	0.47
bm25-none	0.14	0.11	0.26	0.13	0.10	0.25	0.30	0.30	0.54
ql-none	0.13	0.10	0.25	0.13	0.10	0.25	0.31	0.32	0.55
ql-rm	0.08	0.05	0.24	0.08	0.06	0.20	0.23	0.24	0.47
bm25-rm	0.11	0.08	0.24	0.11	0.09	0.24	0.29	0.29	0.52

11 Evaluation on TREC News

We participate in the NEWS track entity ranking task. As no training data was provided, we participate with with three variations of our low-level entity retrieval features as described in Section 3.2.

In this task, we are given a set of entities that we have to re-rank by importance for the article. Our approach is to use a part of the given news article (title or first paragraph) to construct a query. This query is used to retrieve entities from the entity index built from pages made available in CAR's allBut-Benchmark collection. We use BM25 with a whitelist to retrieve a re-ranking Wiki pages and rank the set of given entities according to their pages.

- UNH-TitleBm25: Using the article title as a query, use BM25 to retrieve from the page index.
- UNH-ParaBM25: Using the first paragraph (or at least 200 characters) of the news article's content as a query, use BM25 to retrieve from the page index.
- UNH-ParaBM25Ecm: Using the first paragraph of the news article as

Table 4: Performance of entity re-ranking methods on NEWS. $^{\triangledown}$marks methods with significant difference to best performing method according to a paired-t-test with $\alpha = 0.05$.

Method	**NEWS**		
	MAP	**RPREC**	**NDCG@5**
UNH-ParaBm25	0.82 ± 0.03	0.72 ± 0.04	0.74 ± 0.03
UNH-TitleBm25	0.81 ± 0.03	0.71 ± 0.04	0.72 ± 0.03
UNH-ParaBm25Ecm	$0.71 \pm 0.04^{\triangledown}$	$0.63 \pm 0.05^{\triangledown}$	$0.55 \pm 0.04^{\triangledown}$
median run	N/A	0.66	0.62
best run	N/A	0.76	0.82

query to retrieve CAR paragraphs. The ecm entitt ranking is derived from these paragraphs (using entity links provided by CAR organizers).

The results are presented in Table 4. We see that retrieving entities with the first paragraph (no ecm expansion) works best. For reference we include median and best run as provided by the NEWS organizers. Our best run performs above the median and slightly below the best run.

12 Conclusion

In this notebook we experiment with methods of document retrieval that utilize the context of entity links for passage and entity retrieval.

In the case of CAR entity retrieval, we use the context in which an entity occurs in a passage as a means of indicating the relevance of an entity with respect to a query. This results in a significant improvement over just using features that score the relevance of an entity using fielded queries (such as the title, inlinks, or outlinks fields associated with a Wikipedia entity).

While we were unable to show a similar improvement by retrieving passages using information pertaining to the entities it contains, we conclude that part of the reason for this not working was because entities are less "specific' indicators of relevance than passages. For example, while the entity "United States" may be relevant to a query, this does not necessarily indicate that all passages that contain the entity "United States" are relevant. This suggests potential future work in understanding which entities are most important (salient) in indicating the relevance of passages that contain them. We show that under the right conditions (e.g., Lucene with English stemmer) a combination of traditional retrieval methods with learning to rank results in best performance.

On the NEWS entity ranking task we only retrieved entities from a Wikipedia dump, as no training data was provided. We see that using the first paragraph of the news article is as good as its title.

In comparison to other participating teams in CAR and NEWS track, our methods placed either best or second-best.

References

[1] Laura Dietz, , Ben Gamari, and Jeffrey Dalton. Trec car 2.1: A data set for complex answer retrieval, 2018.

[2] Sean MacAvaney, Andrew Yates, Arman Cohan, Luca Soldaini, Kai Hui, Nazli Goharian, and Ophir Frieder. Overcoming low-utility facets for complex answer retrieval. *Information Retrieval Journal*, pages 1–24, 2018.

[3] Victor Lavrenko and W Bruce Croft. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. ACM, 2017.

[4] Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 365–374. ACM, 2014.

[5] Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.