UMass at TREC 2017 Common Core Track

Qingyao Ai, Hamed Zamani, Stephen Harding, Shahrzad Naseri,
James Allan and W. Bruce Croft

Center for Intelligent Information Retrieval
College of Information and Computer Sciences
University of Massachusetts Amherst
{aiqy,zamani,harding,shnaseri,allan,croft}@cs.umass.edu

1 Introduction

This is an overview of University of Massachusetts efforts in providing document retrieval run submissions for the TREC Common Core Track with the goal of using newly developed techniques in retrieval and ranking to provide many new documents for relevance judgments. It is hoped these new techniques will reveal new documents not seen via traditional techniques, that will increase the numbers of relevant judged documents for the research collection.

2 Tools

We used the Galago search engine, freely available through the Lemur Project (`http://www.lemurproject.org`) with source code available through SourceForge (`https://sourceforge.net/p/lemur/galago/`). This application provides a framework for search engine research and has a very efficient, distributed indexing capability especially useful for large text collections.

The entire LDC2008T19 collection consisting of documents from New York Times articles from 1987 through 2007 was indexed with many different text fields defined to enable very focused query definitions if needed.

The Galago search engine provides the query operators used in defining the search models. These operators include query likelihood, RM3 (relevance model), SDM (sequential dependence model), WSDM (weighted sequential dependence model), Okapi BM25, and PL2 weighting models. Only terms from topic titles were used.

Some of our non-baseline submissions made use of MART and LambdaMART learning to rank models implemented in RankLib (`https://sourceforge.net/p/lemur/RankLib-2.8/`), another Lemur Project library providing several learning to rank models. Additionally, we used Tensorflow (`https://www.tensorflow.org/`) to implement our DIRE model (see Section 5).

For the models that use pre-trained word embedding vectors, we used the vectors learned by GloVe (`https://nlp.stanford.edu/projects/glove/`) [8].

3 Baseline Submissions

Two search models were used in producing the baseline results: Sequential Dependence Model [7] and Relevance Model [5] which are briefly described below.

1. `umass_baselnsdm`: A sequential dependence model (SDM) using the Galago `#sdm` query operator to transform raw query text into a query form reflecting the Sequential Dependence Model. This model assumes dependencies between adjacent query terms. The resulting query form contains three components consisting of unigram, ordered distance and unordered distance operations, each with differing weights that may be specified by the user. Default weights were used for the baseline SDM queries consisting of 0.8 for unigrams, 0.15 for ordered distance and 0.05 for unordered window of query terms. No additional resources were used in producing this baseline and the process was entirely automatic. This model was described in more detail by Metzler and Croft [7].
2. `umass_baselnrm`: A relevance model (RM3) using Galagos `#rm` operator. This is an implementation of a pseudo-relevance feedback process based on language models producing a specified number of query expansion terms from a specified number of top ranked documents returned by an initial query. A second pass search is performed using the original query with a specified weight combined with the derived new expansion terms to obtain the final ranked document returns. Our baseline runs used a default original query weight of 0.25 using the top 10 terms from each of the top 10 originally returned documents. No additional resources were used in producing this baseline and the process was entirely automatic. This model was described in more detail by Lavrenko and Croft [5] as well as Abdul-Jaleel et al. [1].

4 Determining Run Submission Priorities

We focused on retrieved document “uniqueness” in deciding what priority each submission would have with the objective of submitting runs with the most unseen or unique document IDs for later judgment. This involved the following, iterative process:

- Combine top 10 ranked documents from our two baseline runs as the initial pool of document IDs to compare our submission runs against.
- Determine a “uniqueness score” for each submission against this initial pool of document IDs. This score was simply the number of top 10 document IDs in a submission that do not appear in the comparison pool.
- Assign the next available priority level to the current best unique score document.
- Add the top 10 ranked documents from this submission to the comparison pool.
- Continue determining a uniqueness score using this new comparison pool of document IDs to the remaining, not yet prioritized submissions.

- Continue until all submissions have been assigned a priority.
- Submit the runs to NIST in priority order.

Uniqueness determinations were applied only using the 50 NIST topics and not the full 250 topic set.

5 UMass Submissions

The following is a list of UMass submissions (other than the baselines provided as a service to the track) to the Common Core Track. They totalled 10 submissions with three for the phase 1 deadline and seven more for the phase 2 deadline. The runs are ordered by the submission priority (see Section 4). In the following, we briefly describe the method produced each submission.

1. `umass_direlmnvs`: A Deep Interaction Reranking (DIRE) model. It takes the results from the ranked list produced by a learning-to-rank algorithm and reranks them according to the local feature distribution extracted from the list. Here we used the results from LambdaMART trained on Robust04 without validations (`umass_letor_lm`) as the input for the DIRE model and reranked the top 60 documents.
2. `umass_direlm`: A Deep Interaction Reranking (DIRE) model. All settings were same as the `umass_direlmnvs` submission except we used results from the LambdaMART with a validation set (`umass_letor_lmn`) as the inputs for the DIRE model.
3. `umass_diremart`: A Deep Interaction Reranking (DIRE) model. All settings were same as the `umass_direlm` submission except we used results from MART trained on Robust04 as the input to the DIRE model.
4. `umass_maxpas150`: A passage-based retrieval model based on the language modeling approach [6]. The model uses a sliding window with length N and step size N/2 to extract passages from a document. We used N=150 to extract passages and computed the language modeling scores between the query and each of the passages. The documents were ranked by the highest score of its passages. Our language modeling approach is the same as the one introduced by Huston and Croft [4] (settings for Robust04).
5. `umass_maxpas50`: A passage-based retrieval model based on the language modeling approach [6]. All settings were same with as the `umass_maxpas150` submission except we used N=50.
6. `umass_letor_lm`: A learning-to-rank model based on LamdaMART [2]. We used the 250 topic titles with annotations from Robust04 as our training data. Our ranking features include query features (the sum, max, arithmetic/harmonic/geometric means and the standard deviation of the inverse document/corpus frequency and the clarity score [10] for each query term) and model features (the scores from BM25, PL2, LM, SDM, WSDM, RM3, MaxPassage50 and MaxPassage150; the setting for model features is similar to the one used by Liu and Croft [6] as well as Huston and Croft [4]). All features were normalized before the training process. We tuned the tree

number from 1000 to 3000, leaf number from 10 to 30, learning rate from 0.1 to 0.3, threshold candidates from 256 to 768 and randomly selected 10% of the data to validate the performance of different models. We reported the results of the best model on the validation set.

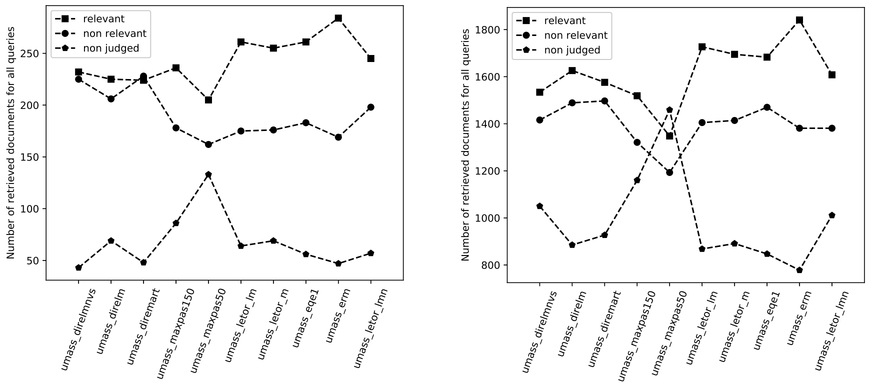
7. `umass_letor_m`: A learning-to-rank model based on MART [3]. All settings were same as with the `umass_letor_lm` submission except we used MART as the learning algorithm.
8. `umass-eqe1`: Embedding-based Query Expansion (EQE), a query expansion model using word embeddings. The underlying assumption of this model is "conditional independence of query terms". We used the word embedding vectors learned by GloVe [8] on the Wikipedia dump 2014 plus Gigawords 5. The embedding dimensionality was set to 300.We linearly interpolated the original query with the expansion terms (the number of expansion terms was set to 10). The interpolation coefficient was set to 0.5. This model was described in more detail by Zamani and Croft [9] (see Section 3.2).
9. `umass_erm`: A pseudo-relevance feedback approach based on word embedding vectors, called Embedding-based Relevance Model (ERM). This model uses both embedding similarities and the information from the top-retrieved documents for each query. We use the same pre-trained embedding vectors as those used in `umass-eqe1`. We used top 10 retrieved documents and expanded the original query using 10 terms. The parameters α and β are both set to 0.5. This model was described in more detail by Zamani and Croft [9] (see Section 3.3).
10. `umass_letor_lmn`: A learning-to-rank model based on LamdaMART [3]. All settings were same as with the `umass_letor_lm` submission except we did not sample the training data to form a validation set. Instead, we used the performance on the training data to select the best model.

6 Analysis

Fig. 1 plots the number of relevant, non-relevant, and unjudged documents for the top 10 (Fig. 1a) and 50 (Fig. 1b) retrieved documents per submission run. As depicted in the figures, `umass_maxpas50` retrieved the highest number of unjudged documents, hence is more likely to retrieve documents that cannot be retrieved by conventional retrieval models. Interestingly, there are few unjudged documents among the top 10 documents retrieved by `umass_direlmnvs`, while it substantially increases when it comes to the top 50 documents. Furthermore, `umass_erm` retrieves the highest number of relevant documents in both cases.

7 Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1160894. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.



(a) Top 10 retrieved documents

(b) Top 50 retrieved documents

Fig. 1: Number of relevant, non-relevant, and unjudged retrieved documents for each submission run.

References

1. N. Abdul-jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M. D. Smucker, T. Strohman, H. Turtle, and C. Wade. Umass at trec 2004: Novelty and hard. In *Proceedings of the 2004 Text REtrieval Conference*, TREC '04, 2004.
2. C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, Microsoft, June 2010.
3. J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.
4. S. Huston and W. B. Croft. Parameters learned in the comparison of retrieval models using term dependencies. Technical report, University of Massachusetts Amherst, 2014.
5. V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New York, NY, USA, 2001. ACM.
6. X. Liu and W. B. Croft. Passage retrieval based on language models. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, CIKM '02, pages 375–382, New York, NY, USA, 2002. ACM.
7. D. Metzler and W. B. 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*, SIGIR '05, pages 472–479, New York, NY, USA, 2005. ACM.
8. J. Pennington, R. Socher, and C. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, 2014.
9. H. Zamani and W. B. Croft. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 147–156, New York, NY, USA, 2016. ACM.

10. Y. Zhao, F. Scholer, and Y. Tsegay. Effective pre-retrieval query performance prediction using similarity and variability evidence. In *Proceedings of the IR Research, 30th European Conference on Advances in Information Retrieval*, ECIR'08, pages 52–64, Berlin, Heidelberg, 2008. Springer-Verlag.