UMass at TREC 2018: CAR, Common Core and News Tracks

Shahrzad Naseri[1] John Foley[2*] James Allan[1]

[1]College of Information and Computer Sciences
University of Massachusetts Amherst
{shnaseri, allan}@cs.umass.edu

[2]Department of Computer Science
Smith College
jjfoley@smith.edu

Abstract

UMass participated in three TREC tasks in 2018: the TREC CAR, TREC Core tasks and TREC News (Background Linking). In this paper we detail the contents of our submissions and our lessons learned from this year's participation.

1 Introduction

This is an overview of University of Massachusetts efforts in providing document retrieval run submissions for the TREC CAR passage retrieval, TREC Core and TREC News background linking tracks. In TREC CAR, our run investigates the impact of similarity between query and documents embedding vector representation on retrieval accuracy with the goal of alleviating vocabulary mismatch problem. For our TREC core submission, we explored more efficient variations on a traditional term dependency model to see if there are detectable differences in new document pools. We also explored some of our efficiency gains in a document feedback context within the TREC News background linking task.

We demonstrate our approaches in retrieval for TREC CAR, TREC Core and TREC News in Section 2, Section 3 and Section 4.

2 TREC CAR

In this year's TREC CAR track, we aim to investigate the long standing vocabulary mismatch problem for the hierarchical complex topics of CAR collection. We address the task of passage retrieval and retrieval from a passage collection exacerbates the vocabulary mismatch problem. This is because of the extensive use of language variety and abbreviations in related series of passages. Neural ranking models [8, 11, 12] tackle this problem by learning a global dense word embedding representation based on the adjacent terms in the context. Using the learned embedding vector, we then can find the similar terms in the multi-dimensional embedding space to a given term. In this work, since CAR is an entity centric collection, we learn a joint word-entity embedding representation on the Wikipedia corpus and represent query and the paragraphs based on their corresponding embedding vector.

2.1 Indexing TREC CAR Passages

We index the paragraphs in TREC CAR corpus using Galago search engine with the link and the paragraph's text fields. Queries' stop words are removed using the 418 INQUERY stopwords list. Stemming is performed using the built-in Krovetz stemmer.

2.2 Learning Joint Word-Entity Embeddings

Motivated from the concept of vocabulary mismatch problem, we learn a joint entity-word embedding following the approach presented by Ni et al. [13]. We learn a low dimensional vector representation for entities and words based on Mikolov Skip-gram model [10] using term co-occurrence information within a

*Work done while at UMass

text. Each entity mention specified by its link is considered as a single "term". The Skip-gram model aims to maximize the probability of current term based on its surrounding terms using a neural network.

The following excerpt shows the transformation of text with entity mentions using special placeholders for each mention:

> A hybrid electric vehicle (HEV) is a type of `hybrid_vehicle` that combines a conventional `internal_combustion_engine` (ICE) system with an electric propulsion system (hybrid_vehicle_drivetrain). The presence of the electric powertrain is intended to achieve either better `fuel_economy` than a `conventional_vehicle` or better performance.

2.3 Embedding-based Passage Ranking

Each topic, i.e. query, in CAR collection is consist of three subtopics. As an example, the query [Antibiotic use in livestock/Use in different livestock/In swine production] has "Antibiotic use in livestock" as root (R), "Use in different livestock" as intermediate (I) and "In swine production" as leaf (L) subtopics. To formulate a query, we use the concatenation of R, I and L. We retrieve a set of documents with three baseline methods: SDM (Sequential Dependence Model), RM3 and query likelihood with the subtopic combinations of R-L, R-I-L and R-I-L, respectively.

Furthermore, we represent each document and query based on their entity embeddings. Each document is represented by average of entities' embedding vector in the document. Each query has a fine-grained subtopic representations as well as a complete topic representation. To be more specific, we represent a query by the average embedding vectors of entities only in Root (R), entities only in Leaf (L) as well as all of the entities in the topic.

We use cosine similarity between the document vector representation and each query representation as features for a LambdaMart learning-to-rank model. Moreover, the retrieval scores from the base retrieval model are also added as features to the learning-to-rank model.

2.4 Run Submitted

We tune the retrieval hyper-parameters on the `benchmarkY1-train.v2.0` training data using grid search. For the Query Likelihood model the smoothing parameter $\mu = 400$ gives the optimal result. For the Sequential Dependence Model (SDM) baseline $mu = 1200$, $uww = 0.02$, $odw = 0.10$, and $uniw = 0.82$ are the best parameters.

For pseudo relevance feedback baseline, we use SDM model as the baseline retrieval method. The expansion parameters are tuned similarly using grid search and we find that 10 expansion documents with 20 feedback terms and an interpolation weight of 0.8 is most effective. For document entity annotations we use the existing links provided in Wikipedia. Furthermore, for query annotation, we use the open-source state-of-the-art SMAPH entity linker [1] for each query's subtopic.

In our experiment all parameter tuning as well as learning the learning-to-rank model was performed on `benchmarkY1-train.v2.0`. `benchmarkY1-test-public.v2.0` data was used as a validation set during training the learning-to-rank model. We use LambdaMart as our learning-to-rank model.

The result for our submitted run is showed in Table 1. As results suggest, the best run is the `rl-sdm` run, and our global embedding vector similarity features were unable to beat the strong `rl-sdm` baseline.

Table 1: Paragraph retrieval on `benchmarkY2test` Manual

Model	MAP	R-Prec	NDCG
`umass_entityEmbedLambdaMart`	0.2792	0.2803	0.5223
`rl-sdm`	0.3542	0.3488	0.5769
`ril-rm3`	0.2482	0.2580	0.4878

3 TREC Core

For the TREC Core task, we chose to explore some approximations to traditional term dependency models.

[1] `https://github.com/marcocor/smaph`

3.1 Indexing WAPO Articles

We indexed the Wapo corpus provided for both news and core tasks with the following fields (when available): `published.date`, `url`, `kind`, `title`, `author`, `kicker`, and we constructed a body from the text of (HTML tags were removed with JSoup[2]) the JSON `paragraph` "blobs".

3.2 Background: Dependency Models

Many models try to incorporate textual information beyond simple unigrams. One of the most commonly used models is the sequential dependence model (SDM) described by Metzler and Croft [9]. Variations of this model are still relevant to many applications today, such as entity retrieval [17].

Although this model is described as a markov random field, Dietz and Foley showed that this is equivalent to a handful of other formulations, particularly generative models [3].

Huston and Croft surveyed a wide variety of models that integrate dependency into query-document scoring [5]. In a later, unpublished technical report, they compare the many different formulations of so-called `UnorderedWindow` dependencies (broken down by what kinds of overlap are allowed) and determined that there was no observable statistical difference between common formulations [6].

Surprisingly, there is little research focusing on the efficiency and effectiveness tradeoffs between feature functions chosen in dependency models. Huston's thesis is a rather complete discussion of the tradeoffs between indexing and computing dependencies on the fly, and includes discussion of a sketching technique for more cheaply estimating these statistics and counts [4].

3.3 Cheap Sequential Dependency Modeling

In this subsection, we describe a "bag of tricks" for cutting down the execution time of traditional SDM [9] queries. Our evaluation will show that these tricks have limited impact on overall accuracy, making our CSDM formulation a practical alternative version to this classical model.

In the sequential dependence model, term dependencies are assumed between adjacent terms in a query. Unigrams are also kept as the primary ranking of documents, and dependencies are given lower weights in comparison.

3.3.1 Min-Estimate Statistics

Most modern retrieval models have some kind of probabilistic interpretation and specify either the collection frequency of a phrase or the document frequency of a phrase as a necessary component for normalizing values. In most systems, like Galago this means that a dependency query becomes two evaluations for the underlying engine: once to compute the collection-wide frequencies and once to compute the actual document values.

We propose a simple alternative to this normalizer that eliminates one pass of batch processing. Instead of calculating the exact value (which involves visiting every document): $F(w_i, w_j, C) = \sum_{d \in C} F(w_i, w_j, d)$, we leverage the pre-stored unigram statistics of $F(w_i, C)$ and $F(w_j, C)$ to estimate $\hat{F}$. Since traditional dependencies – both ordered and unordered – involve the intersection of posting lists, we can estimate their statistics as upper-bounded by the minimum: $\hat{F}(w_i, w_j, C) = \min(F(w_i, C), F(w_j, C))$. This minimum as upper-bound is intuitive: the phrase "garden vegetables" can occur no more than "garden" or "vegetables" occur independently.

Since it is an upper-bound, it works fine as a normalizer (no estimated probabilities above 1.0).

3.3.2 Stopwords only in Dependencies

One common efficiency trick in any query processing is to elide stopwords. However, famous TREC queries such as "to be or not to be" encourage system designers to leave stopwords in their indexes and and their batch query submission systems.

If we elide stopwords from the unigram representation, we handle queries like "to be or not to be" as a sequence of dependencies where intersection of posting lists can be used to limit candidate scoring, e.g., "[D(to be) D(be or) D(or not) D(not to) D(to be)]". While still dramatically cutting down the number of candidates (instead of processing and fully-scoring nearly all documents, we can eliminate some that do not have rarer bigrams like "be or". Once you toss in some query intelligence and feature folding, you can often execute this stopped query much faster [2].

[2] `https://jsoup.org/`

3.3.3 MinCount instead of UnorderedWindow

The core purpose behind window dependencies is to up-weight documents in which terms appear close together. Unfortunately, calculating the count of times two terms occur within a fixed window of each other requires a lot of storage or posting lists.

Instead, we propose looking at the `MinCount` of terms within a document. This provides an upper-bound (once-again) on the true value of any `UnorderedWindow`. Motivated by Huston and Croft's technical report findings [6] that variations of `UnorderedWindow` functions are not that impactful on retrieval, we decided to investigate just how much the actual term proximity mattered, or if what were achieving was mostly just a softer boolean AND of the terms in question.

In the SDM formulation of a query, there are often phrase nodes (e.g., OrderedWindows) and span nodes (the UnorderedWindows). Especially in the presence of the OrderedWindows (there are many good tricks to compute phrases, such as next-word indices [16]), it is unlikely that a first-pass retrieval model is distinguishing many documents based on term occurrence distances.

3.3.4 BM25 instead of Query Likelihood

The final "trick" we consider is that of a BM25-based SDM. While both models are state-of-the-art unigram retrieval models, BM25 has some nice properties: it is purely additive, no expensive logarithm computation is done, and its minimum score is bounded to zero. This means that BM25 often has better performance when used with early termination algorithms such as MaxScore [15] and WAND [1, 14].

3.4 TREC Core Runs Submitted

For these runs our BM25 parameters were the Galago default, our Dirichlet μ was set to be the average document length, and our SDM parameters were set to the commonly-chosen defaults of ($\lambda_u = 0.8, \lambda_{\text{odw}} = 0.15, \lambda_{\text{uww}} = 0.05$).

Since we verified offline that our minimum-based statistics estimation (§3.3.1) makes no discernible impact on other collections, we used it for all of our dependency models.

`umass_ql` This run is a standard query-likelihood model.
(mAP = 0.231, ndcg = 0.516)

`umass_sdm` This run is a standard SDM model.
(mAP = 0.234, ndcg = 0.518)

`umass_bsdm` This run is a BM25-based SDM model.
(mAP = 0.231, ndcg = 0.509)

`umass_cbsdm` This run is a BM25-based SDM model with our MinCount trick added.
(mAP = 0.228, ndcg = 0.508)

3.5 TREC Core Results Discussion

We proposed to explore features where we did not expect much change in effectiveness for a large change in efficiency. Because there were no queries available on this dataset, we were unable to tune our query likelihood and BM25 parameters. This means that our best hypothesis between differences between runs is still that our BM25 default parameters are not as good as our QL parameters. We look forward to the release of the track-collected judgments in order to investigate this direction further under a cross-fold validation setup.

We performed significance testing to determine whether the differences observed in mean average precision (mAP) were different within queries. We find that no distinction can be claimed even though there are differences at (± 0.005). This suggests that on this dataset, with this power, with default parameters on each model, our "bag of tricks" for making SDM faster observes some differences, but with fairly high odds, (30%-40%) it is actually due just to random chance. We plan to investigate parameter tuning differences and multiple datasets in future work.

4 TREC News BG Linking

For the TREC News Background Linking task, we re-used our indices built for our TREC Core submission. The procedure for creating the indexes is described in Section 3.1.

		baseline	
		umass_ql	umass_sdm
treatment	umass_ql	1.000	0.384
	umass_sdm	0.616	1.000
	umass_bsdm	0.479	0.381
	umass_cbsdm	0.380	0.294

Table 2: One-sided p-value computation for whether runs are different: low p-values (typically $p < 0.05$) would indicate true differences. We used the pairwise randomization test derived from Galago in order to compute run differences

4.1 Document Query Generation

We began by computing a traditional relevance model [7], which provides a mapping of unigrams to probabilities, $P(w|R_D)$, where the set of relevant documents is just the input query document: $P(w|D_Q)$, excluding the `inquery` stopwords provided with Galago[3].

For all pairs of terms w_i, w_j we then compute the minimum distance between them as they occur in the original source document: mindist(w_i, w_j). We extracted a set of dependencies for which mindist$(w_i, w_j) \leq M$, where the minimum distance was left as a parameter. We weighted these dependency parameters by their importance: the geometric mean of their probabilities $P(w_i|D_Q) \cdot P(w_j|D_Q)$. We used the full set of detected dependencies and combined it linearly with the traditional relevance model expression (capped to a parameter k terms). We used Dirichlet smoothing for all probability computations.

4.2 Result Filtering

Duplicate detection was a challenge this year, so we de-duplicated documents by title (choosing the most recent publication date for any conflicts) and rejected all documents without a title from ranking – feeling that documents are not useful for background research if they cannot be summarized concisely.

We also enforced that documents retrieved were published before the query document, using a streaming model for our evaluation. Following the track guidelines, we also excluded documents whose kickers were: "Opinion(s)", "Letters to the Editor", or "The Post's View".

4.3 Runs Submitted

We briefly tuned our μ, N, M and λ parameters on a simulated task over Robust04, because it is also a News corpus. Because we did not learn different parameters for different queries, we did not consider this run to be using prior knowledge. The parameters selected were quite close to default parameters used in the Galago search engine. We had $N = 50$ feedback terms, a window detection threshold of $M = 8$, and a lambda between dependencies of $\lambda = 0.8$.

Our TREC submissions here had no significant differences, for a variety of parameter selection reasons. We hope to explore window dependencies extraction in further Background linking submissions.

`umass_rm` This run is a standard relevance model.

`umass_rdm` This run was an attempt to mix extracted unordered window dependencies into a standard `umass_rm` model, but in practice these dependencies barely changed the ranking, and did not affect measures, despite searching for a bug.

`umass_cbrdm` This run was actually just an RM model with BM25 scoring functions, due to a processing bug.

Run Name	Scorer	NDCG@5	NDCG@1000
umass_rm	dirichlet	0.4157	0.4658
umass_rdm	dirichlet	0.4157	0.4658
umass_cbrdm	bm25	0.4173	0.4688

Table 3: Runs submitted to TREC.

[3] `http://lemurproject.org/galago.php`

4.4 Results Discussion

While we intended for term dependency extraction from relevance documents to have a bigger impact (due to weighting and relative lack of importance they did not) our results here do suggest that our cheaper dependencies do not have a significant impact and that using BM25 rather than query likelihood does not result in a loss of performance on an additional task.

4.4.1 Time Filtering

We submitted runs that pruned all documents published after the query document. Documents missing a publication time were considered to be older than any others and were therefore included. Although this was not part of the final ruleset, it actually significantly improved results, at least in terms of precision! (`umass_rm` achieves lower performance without time filtering: NDCG@5=3.556 vs. NCDG@5=0.4157).

In future submissions to TREC News, we hope to explore this relationship between time and relevance more deeply.

5 Conclusion

We present our approach in TREC CAR, TREC Core and TREC News background-linking tasks. SDM is a powerful baseline: our TREC CAR submission was unable to beat it (with more features) and we have preliminary evidence to suggest that simpler features lead to equivalent retrieval power. In TREC Core, we found that our approximations to standard SDM functions caused no significant drops in performance, and in TREC News we found that background articles being before a query article was a highly useful feature. We plan to study our models in more detail in the future.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval. 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.

References

[1] Broder, A.Z., Carmel, D., Herscovici, M., Soffer, A., Zien, J.: Efficient query evaluation using a two-level retrieval process. In: Proceedings of the twelfth international conference on Information and knowledge management. pp. 426–434. ACM (2003)

[2] Cartright, M.A., Allan, J.: Efficiency optimizations for interpolating subqueries. In: Proceedings of the 20th ACM Conference on Information and Knowledge Management. pp. 297–306 (2011)

[3] Dietz, L., Foley, J.: On the equivalence of generative and discriminative formulations of the sequential dependence model. arXiv preprint arXiv:1805.00152 (2018)

[4] Huston, S.: Indexing Proximity-based Dependencies for Information Retrieval. Ir, University of Massachusetts Amherst (September 2013)

[5] Huston, S., Croft, W.B.: A comparison of retrieval models using term dependencies. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. pp. 111–120. ACM (2014)

[6] Huston, S., Croft, W.B.: Window extraction for information retrieval. IR IR-976, University of Massachusetts (2014)

[7] Lavrenko, V., Croft, W.B.: Relevance based language models. In: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 120–127. ACM (2001)

[8] MacAvaney, S., Yates, A., Cohan, A., Soldaini, L., Hui, K., Goharian, N., Frieder, O.: Characterizing question facets for complex answer retrieval. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 1205–1208. ACM (2018)

[9] Metzler, D., Croft, W.B.: 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. pp. 472–479. ACM (2005)

[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)

[11] Mitra, B., Diaz, F., Craswell, N.: Learning to match using local and distributed representations of text for web search. In: Proceedings of the 26th International Conference on World Wide Web. pp. 1291–1299. WWW '17, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland (2017)

[12] Nanni, F., Mitra, B., Magnusson, M., Dietz, L.: Benchmark for complex answer retrieval. In: Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval. pp. 293–296. ICTIR '17, ACM, New York, NY, USA (2017)

[13] Ni, Y., Xu, Q.K., Cao, F., Mass, Y., Sheinwald, D., Zhu, H.J., Cao, S.S.: Semantic documents relatedness using concept graph representation. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining. pp. 635–644. ACM (2016)

[14] Petri, M., Culpepper, J.S., Moffat, A.: Exploring the magic of wand. In: Proceedings of the 18th Australasian Document Computing Symposium. pp. 58–65. ACM (2013)

[15] Turtle, H., Flood, J.: Query evaluation: strategies and optimizations. Information Processing & Management **31**(6), 831–850 (1995)

[16] Williams, H.E., Zobel, J., Anderson, P.: What's next? index structures for efficient phrase querying. In: Australasian Database Conference. pp. 141–152. Citeseer (1999)

[17] Zhiltsov, N., Kotov, A., Nikolaev, F.: Fielded sequential dependence model for ad-hoc entity retrieval in the web of data. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 253–262. ACM (2015)