Smith at TREC2019: Learning to Rank Background Articles with Poetry Categories and Keyphrase Extraction

John Foley, Ananda Montoly and Mayeline Peña
{jjfoley, amontoly, mpena}@smith.edu

Department of Computer Science
Smith College

Abstract

Smith College participated in the TREC News Background Linking task in 2019. We constructed a linear learning to rank model trained on the 2018 data and submitted runs that included features derived from ongoing research into automatic poetry understanding.

In this notebook paper we detail the contents of our submissions and our lessons learned from this year's participation.

1 Introduction

The TREC News Task is composed of two core tasks: Background Linking and Entity Linking. We only participated in Background Linking this year. Given a single news document, the goal is to rank other documents from the rest of the collection with respect to how useful they would be for background reading, in suggestion to a user.

In addition to devising a learning-to-rank approach based upon more traditional features, we leveraged some concurrent research we have been doing on poetry data on this news corpora. Recently, Foley's dissertation introduced a large public collection of poetry extracted from internet archive books [13]. We had two projects working with this dataset over this summer and we converted each of these to features in our approach.

We submitted four runs to the background-linking task, as described in the next section. We describe our indexing and scoring process in Section 2, the new learning to rank tool we used (FastRank) and our baseline features in Section 3, and our poetry research features that made it into our submission in Section 4. Finally we have a brief discussion of our 2019 results in Section 5

1.1 Runs Submitted

`smith_base` This contains the baseline LTR features, described in Table 1.

`smith_poetry` This contains our baseline model with similarity based on detected poetry categories in the news articles. This model is described in Section 4.1.

`smith_keywords` This contains the baseline model with additional features based on our keyword importance model. This model is described in Section 4.2.

`smith_full` This contains a learning-to-rank model over all the features we studied. This combines the previous three models into one.

In Proceedings of the TREC 2019 Workshop.

2 News-Specific Processing

We first create an index using the `irene` software[1] that is a combination of the `inquery` programming language [8, 28, 9, 18] with the Lucene document indexing system [23].

2.1 Indexing WAPO Articles

We indexed the Washington Post 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".

2.2 Result Filtering

Duplicate detection was part of the challenge of working with Washington Post data 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".

3 Learning to Rank Baseline

Since this is the second year of TREC News, this year there was training data available for a supervised approach. All of our runs fall into this category, using the 50 queries from News 2018 in five-fold cross-validation for learning.

3.1 FastRank: An Enhanced Coordinate Ascent

One of the most effective and simplest learning to rank models for small datasets is the Coordinate Ascent model [21] because of its ability to directly optimize IR evaluation measures. This model is stochastic and searches parameter space one feature at a time (with acceleration) to construct a linear weighting of features. It proceeds until no improvements are available with a given tolerance.

For our learning to rank experiments we rewrote the implementation available in Ranklib [11] in Rust (to improve efficiency), to simplify the code, and to add a few small enhancements:

1. Random initialization: Ranklib initializes its starting weight vectors to be $1/D$ where D is the number of features; thus assuming each feature is equally important and in the positive sense. Since the algorithm uses random restarts, to avoid local minima, it also makes sense to randomly initialize each weight vector.

2. Feature selection: in addition to moving features from their current values based on an accelerating step-size, we also consider removing each feature at every point during the search: that is, setting the feature value to zero. This prevents model performance from dropping precipitously when a poor feature is introduced if it can observe an improvement from eliding it.

An early release of our new learning to rank tool is available[3] and was announced recently via blog post [14]. In addition to these algorithmic improvements of Ranklib, this tool also has deterministic seeding, support for better error messages while loading improperly formatted input files, and support for compressed files.

3.2 Learning to Rank Features

Our learning to rank features fall into three categories, and are presented in Table 1.

Retrieval Models The first category of features are standard retrieval models. We turn the original article into a query and execute the given retrieval model against all pooled results.

[1] `https://github.com/jjfiv/irene`
[2] `https://jsoup.org/`
[3] `https://github.com/jjfiv/fastrank`

Reverse Models Rather than treating the original article as a query, we can go in reverse. That is, we consider the candidate article as a query and the original article as the document to be scored. Most modern retrieval models are non-symmetric so this provides additional information about how similar the query document and the candidate documents are.

Quality Features The last category of features we describe are quality & metadata features. They are query-independent and are meant to be a kind of filter on the types of documents that are valuable to recommend to a user. In addition to some traditional quality features [3], and two date features to allow the model to prioritize past articles over future articles (or vice versa) we also generated a clickbait feature for WaPo articles based on their title (See Section 3.2.1 for more).

Group	Feature	Description
Retrieval Models	title-sdm	The title of the query article as a SDM query [20]
	title-ql	The title of the query article as a Query Likelihood query [24, 32]
	rm-n10	A relevance model (RM) [17] built with 10 expansion terms.
	rm-n50	RM with 50 terms.
	rm-n100	RM with 100 terms.
	bm25-rm-n10	A relevance model (RM) [17] built with 10 expansion terms weighted with BM25 [27].
	bm25-rm-n50	RM with 50 terms weighted with BM25.
	bm25-rm-n100	RM with 100 terms weighted with BM25.
Reverse Models	body-ql	Body of query document as an unweighted unigram query.
	first-para-ql	First paragraph of query document as an unweighted unigram query.
	first-para-sdm	First paragraph of query document as an SDM query.
	title-ql-rev	Title(candidate document) as a query against the query document.
	bm25-rm-n10-rev	RM with 50 terms weighted with BM25 (cand to query).
	bm25-rm-n50-rev	RM with 50 terms weighted with BM25 (cand to query).
	bm25-rm-n100-rev	RM with 100 terms weighted with BM25 (cand to query).
Quality Features	entropy	Entropy of candidate document.
	body-len	Length of candidate document.
	date-cmp	Three values: candidate published $(<, =, >)$ the query document.
	date-sub	The difference in publication times.
	clickbait-proba	Clickbait-probability of candidate title. (§3.2.1)

Table 1: List of Learning to Rank Features

3.2.1 Clickbait Classification

In news, clickbait refers to headlines that are intentionally vague, sensational, or otherwise provocative in order to gather clicks. While there has been a variety of work on clickbait detection [26, 1, 10] and even a challenge [25], one of the easiset datasets to obtain is that of Chakraborty et al.,[10], which is available in plain text on Github[4].

Although most works define and extract features manually, (Chakraborty et al. use a SVM to get 93% accuracy) instead we used a simple neural network approach via the `fastText` tool[5] for supervised learning of word embeddings for the classification task [15]. This tool obtained a P@1 (Accuracy) of 0.959 which was as good or better than the reported SVM accuracy without the need for explicit feature extraction.

We applied this learned `fastText` clickbait model to the tiles of all of our candidate documents with the intuition that better background articles would have less clickbait-y titles.

Results over an early version of the base LTR model on 2018 data suggested a reasonably-sized improvement in NDCG@5 (0.386 to 0.403) using our enhanced Coordinate Ascent model (§3.1).

[4] `https://github.com/bhargaviparanjape/clickbait`

[5] `fasttext supervised -input clickbait.train -output model -dim 32 -lr 0.1 -wordNgrams 4 -minCount 1 -bucket 10000 -epoch 20`

4 Poetry Research for News

In this section, we discuss how we took ongoing research into poetry categorization and keyphrase extraction and applied it to our TREC News background-linking runs.

4.1 Poetry Categorization

Recently, more interest has been given toward extracting datasets of poetry from within other datasets [30, 29, 16, 13]. With Foley's dissertation work releasing 600,000 pages[6] of poetry [13], we were looking into how to categorize this poetry into the categories provided by the `https://poetryfoundation.org` that were previously studied [19]. There are 9 first-level categories and 127 subcategories available in that labeled dataset.

Using some tools from the `scikit-learn` package we trained linear classifiers over TF-IDF representations for the 9 top-level categories, each of which had thousands of poetry labels. Since the top-level categories had enough labels, and TF-IDF is a powerful baseline, these classifiers achieve high performance in both cross-validation and generalization (to Foley's Poetry50K dataset) settings. Results for cross-validation are presented in Table 2 – generalization experiments and labels are not complete.

Category	Number of Labels	AUC
Social Comm.	3050	0.753
Relationships	3050	0.722
Nature	2644	0.797
Arts/Sciences	2270	0.723
Living	3946	0.668
Religion	1049	0.804
Love	1608	0.826
Activities	1461	0.678
Mythology	468	0.716

Table 2: Cross-Validation AUC for Poetry Classification

The top-labels were: "Relationships", "Nature", "Arts/Sciences", "Living", "Religion", "Social Commentaries", "Love", "Activities", and "Mythology". We turned our nine classifiers into a nine-dimensional vector for every Washington Post article. We then added a feature that was the cosine similarity between the query article and a particular background article, as well as query-independent features for each topic.

Article Title	Top-3 Labels
Dave Chappelle accepts the Twain Prize: 'I love my art form. It saved my life.'	Social Comment. (1.6), Arts/Sci. (1.3), Relationships (0.49)
Apples keep you healthy, carrots help your eyes: What science says about such folk remedies	Activities (1.5), Arts/Sci. (0.41), Social Comment. (0.37)

Table 3: Qualitative inspection of labels assigned to articles seem yields fairly positive results, especially for Top-1 labels. (Used non-corpora Washington Post stories; 27 October and 16 October 2019, respectively.)

Some results are presented on both 2018 and 2019 data in Table 5. Because results were competitive using NDCG and this had not been part of last year's pool, we decided to submit the `smith_poetry` run.

4.2 Keyword and Keyphrase Extraction

In poetry, often entities or other important concepts are mentioned without context; this is sometimes used as allusion, to add color, emotion or context to a poem with very few tokens. One of the the open research questions we have is how to identify such allusions and other interesting words. One obvious baseline for such a task is "keyphrase extraction".

TextRank is the classical keyword or keyphrase extraction approach: it is a version of PageRank where nodes are words or phrases and links between them are crafted from co-occurrence data. The SummaNLP library [2] is one such implementation of this approach in Python. Another library is PKE, which contains a wider variety of keyphrase extraction methods [5], which was used in last year's TREC News Track [4].

[6] `https://ciir.cs.umass.edu/downloads/poetry`

The models available in SummaNLP and PKE that we found helpful and usable on both our poetry and news data are as follows:

TextRank is the classic algorithm: pagerank applied to words with links based on co-occurrence [22].

SingleRank is a model that uses text-rank over candidates that are adjacent nouns and adjectives [31].

PositionRank is a model that extends TextRank by biasing keywords to those occurring earlier in the article or document [12].

MultiRank is a model that only allows edges between phrases of different topics [6].

TopicRank is a model that executes SingleRank and then limits selections to the best keyphrase in each "topic" [7].

Extraction Method	Libary	NDCG@5
TextRank	Summa	0.306
TextRank	PKE	0.254
SingleRank	PKE	0.310
PositionRank	PKE	0.300
MultiRank	PKE	0.336
TopicRank	PKE	0.340
BM25-RM-50	Retrieval Model	0.344

Table 4: NDCG@5 of queries generated by keyphrase extraction. Each phrase extracted is executed as a weighted SDM query [20] based on its extraction confidence.

In Table 4 we present the performance of these off-the-shelf approaches to the TREC News Background-Linking task: we extracted keyphrases from each of the query documents and constructed a query that was a weighted combination of SDM [20] models around each extracted keyphrase. These were competitive with a simpler relevance-modeling feature already present in our ensemble, and provided a small gain in cross-validation over the 2018 data when used together.

5 Discussion

Preliminary analysis of our model results on the 2019 challenge suggest that keywords were of more value in the 2019 data than in the 2018 data. See Table 5, below.

Submitted Run	NDCG@5 2019	NDCG@5 2018	NDCG 2019	NDCG 2018
`smith_poetry`	0.555	0.518	0.603	**0.557**
`smith_base`	0.564	**0.526**	0.605	0.555
`smith_keywords`	**0.570**	0.502	0.610	0.553
`smith_full`	0.560	0.518	**0.614**	0.555

Table 5: Evaluation of submitted models on qrel-files. Official Evaluation metric: NDCG@5

We expected our poetry and keyword features to be helpful as we looked at NDCG while training and evaluating our learning-to-rank model. Because of the exponential discounting in NDCG we disregarded the impact of using NDCG@5 vs. NDCG@100 or larger depths, but the numbers in our Table 5 suggest that this was potentially a mistake. When we visualize the data as bar-plots, we can see that these runs are a little more interchangeable and that the rankings are very sensitive to the depth examined. Rankings are also different for the NDCG@100 case for mean and median NDCG (Figure 1, below).

The poetry features appeared to be valuable in the 2018 data (or at least not significantly harmful but they appear to be more of a problem in the 2019 data). That being said, the differences discovered in including poetry features in our news articles are probably not significant. This means that – our results are probably essentially the same across our four runs. It remains to be seen how runs with totally different features are ranked against these, since absolute measures can be misleading and ranking across systems is what is most robust in TREC evaluations.

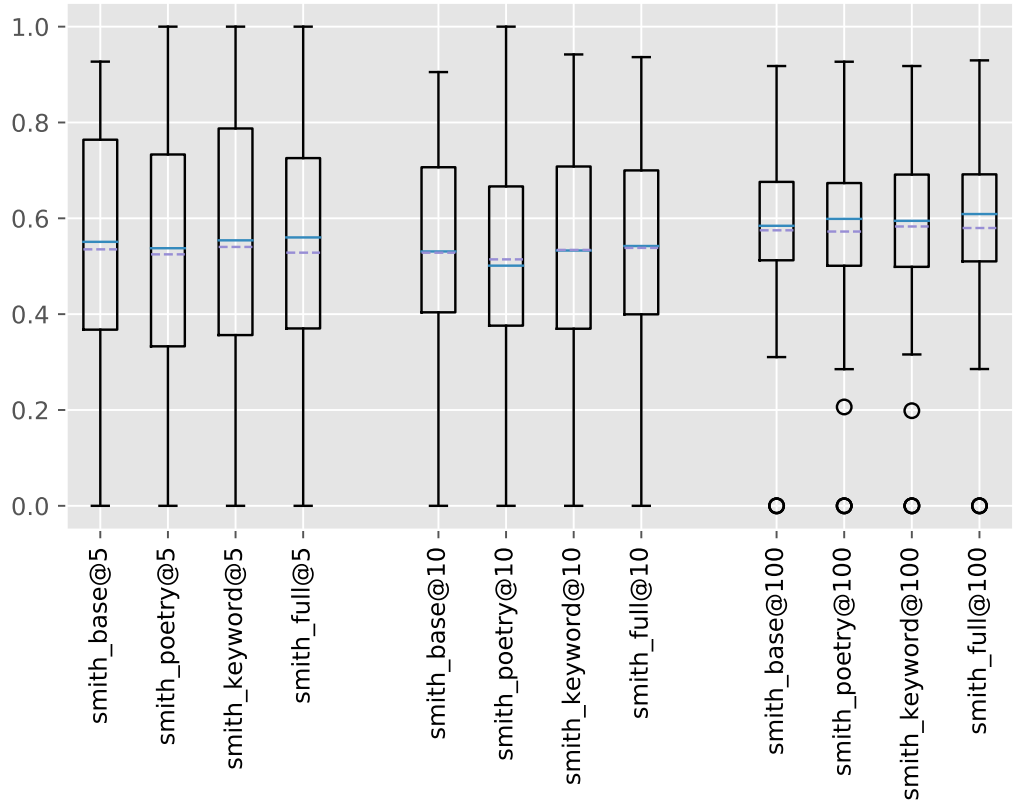

Figure 1: Exploration of Depth on NDCG on system ranking. The ranking of systems changes between a depth of 5, 10 and 100. Clearly unjudged documents will have more of an effect at deeper depths, but recall will be more emphasized in deeper runs.

6 Conclusions

We explored some non-traditional features atop a traditional learning to rank model. Since we tuned our models based on NDCG and not NDCG@5, we did not necessarily emphasize precision as much as we should have during model selection, however, the performance of all our models is highly-overlapping (Figure 1). Surprisingly, keyword extraction methods seem to be more competitive on the 2019 data than on the 2018 data and our poetry features seem to be more helpful for recall than precision; thus not being reflected as clearly in the official measure.

The 2019 dataset is larger than the 2018 dataset and so we plan to experiment with these ideas on the full set of 110 queries which should lead to more statistically significant conclusions.

Acknowledgements

This work was supported in part by the Smith College Committee on Faculty Compensation and Development and the Smith College Science Center's Summer Undergraduate Research Fellowships (SURF). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

[1] Anand, A., Chakraborty, T., and Park, N. (2017). We used neural networks to detect clickbaits: You won't believe what happened next! In *European Conference on Information Retrieval*, pages 541–547. Springer.

[2] Barrios, F., López, F., Argerich, L., and Wachenchauzer, R. (2016). Variations of the similarity function of textrank for automated summarization. *CoRR*, abs/1602.03606.

[3] Bendersky, M., Croft, W. B., and Diao, Y. (2011). Quality-biased ranking of web documents. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 95–104, New York, NY, USA. ACM.

[4] Bimantara, A., Blau, M., Engelhardt, K., Gerwert, J., Gottschalk, T., Lukosz, P., Piri, S., Shaft, N. S., and Berberich, K. (2018). htw saar@ trec 2018 news track. In *TREC*.

[5] Boudin, F. (2016). pke: an open source python-based keyphrase extraction toolkit. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 69–73, Osaka, Japan.

[6] Boudin, F. (2018). Unsupervised keyphrase extraction with multipartite graphs. *arXiv preprint arXiv:1803.08721*.

[7] Bougouin, A., Boudin, F., and Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction.

[8] Callan, J. P., Croft, W. B., and Harding, S. M. (1992). The inquery retrieval system. In *Database and expert systems applications*, pages 78–83. Springer.

[9] Cartright, M.-A., Huston, S., and Feild, H. (2012). Galago: A modular distributed processing and retrieval system. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 25–31.

[10] Chakraborty, A., Paranjape, B., Kakarla, S., and Ganguly, N. (2016). Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16. IEEE.

[11] Dang, V. (2013). Ranklib, `v.2.12`. `https://sourceforge.net/p/lemur/wiki/RankLib`.

[12] Florescu, C. and Caragea, C. (2017). Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1105–1115.

[13] Foley, J. (2019). *Poetry: Identification, Entity Recognition, and Retrieval*. PhD thesis, University of Massachusetts.

[14] John Foley (2019). FastRank alpha release . `https://jjfoley.me/2019/10/11/fastrank-alpha.html`.

[15] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431. Association for Computational Linguistics.

[16] Kilner, K. and Fitch, K. (2017). Searching for My Lady's Bonnet: discovering poetry in the National Library of Australia's newspapers database. *Digital Scholarship in the Humanities*.

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

[18] Lin, J., Crane, M., Trotman, A., Callan, J., Chattopadhyaya, I., Foley, J., Ingersoll, G., Macdonald, C., and Vigna, S. (2016). Toward reproducible baselines: The open-source IR reproducibility challenge. In *European Conference on Information Retrieval*, pages 408–420. Springer.

[19] Lou, A., Inkpen, D., and Tanasescu, C. (2015). Multilabel subject-based classification of poetry. In *The Twenty-Eighth International Flairs Conference*.

[20] Metzler, D. and Croft, W. B. (2005). 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.

[21] Metzler, D. and Croft, W. B. (2007). Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274.

[22] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

[23] Pérez-Iglesias, J., Pérez-Agüera, J. R., Fresno, V., and Feinstein, Y. Z. (2009). Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*.

[24] Ponte, J. M. and Croft, W. B. (1998). *A language modeling approach to information retrieval*. PhD thesis, University of Massachusetts at Amherst.

[25] Potthast, M., Gollub, T., Hagen, M., and Stein, B. (2018). The clickbait challenge 2017: towards a regression model for clickbait strength. *arXiv preprint arXiv:1812.10847*.

[26] Potthast, M., Köpsel, S., Stein, B., and Hagen, M. (2016). Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer.

[27] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. (1995). Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

[28] Strohman, T., Metzler, D., Turtle, H., and Croft, W. B. (2005). Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, volume 2, pages 2–6. Citeseer.

[29] Underwood, T. (2014). Understanding Genre in a Collection of a Million Volumes. Technical report, University of Illinois, Urbana-Champaign.

[30] Underwood, T., Black, M. L., Auvil, L., and Capitanu, B. (2013). Mapping mutable genres in structurally complex volumes. In *IEEE Big Data*, pages 95–103.

[31] Wan, X. and Xiao, J. (2008). Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860.

[32] Zhai, C. and Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.