An Empirical Study of Embedding Features in Learning to Rank

ABSTRACT

This paper explores the possibility of using neural embedding features for enhancing the effectiveness of ad hoc document ranking based on learning to rank models. We have extensively introduced and investigated the effectiveness of features learnt based on word and document embeddings to represent both queries and documents. We employ several learning to rank methods for document ranking using embedding-based features, keyword-based features as well as the interpolation of the embedding-based features with keyword-based features. The results show that embedding features have a synergistic impact on keyword based features and are able to provide statistically significant improvement on harder queries.

ACM Reference format:
. 2016. An Empirical Study of Embedding Features in Learning to Rank. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Learning to rank models have traditionally employed *query dependent* and *query independent* features to rank a set of documents that have been deemed to be relevant to a query. These features include different weighting features such as BM25 [5], term-dependency proximity models such as MRF [7], link analysis features including PageRank and content quality features [1]. However, while recent advances in distributional semantics and more specifically on word and document embeddings have shown impressive performance improvements in many downstream applications [9], there have been few, if any, studies that have systematically explored the development of feaures based on embeddings for learning to rank. The geometric properties exhibited by embeddings representing words and documents enable reasoning about semantic relationships and can hence make them suitable for defining new types of features.

In this paper, we focus on the empirical investigation of the effectiveness of a new set of features that can be defined based on word and document embeddings as well as their interpolation with keyword based features, such as those presented in LETOR 4.0, for improving the performance of ad hoc document ranking. We define features based on embeddings from two cross-cutting aspects: (1) first we consider the *'type of content'* that will be used for building features. Conventional features often rely on the *words* or a subset of the content (e.g., a sentence or paragraph that we refer to as *chunk*), which appear in documents and queries. In addition to words and chunks, we also consider the *entities* identified in the documents and queries. Xiong et al [11] have argued that the use of entities can recognize which part of the query is the most informative; (2) the second aspect that we consider is the *'type of embeddings'* that can be used. We have included both word embedding [8] and document embedding [4] models for building different features.

Based on these two cross-cutting aspects, namely type of content and type of embedding, we have defined five classes of features as shown in Table 1. Briefly stated, our features are *query-dependent* features and hence, we employ the embedding models to form vector representations of words, chunks and entities in both documents and queries that can then be employed for measuring the distance between queries and documents. It should be noted that we do not use entity embeddings [3] to represent entities, but rather use the same word embedding and document embedding models on the abstract of the entities to form an embedding representation for the entities. Our future work will investigate the application of entity embeddings.

As an additional set of features, we merge the LETOR 4.0 features with each of the proposed embedding feature sets, which we refer to as feature *interpolation* and train rankers based on the merged set of (intepolated) features. Our analysis shows that embedding features, especially when interpolated with keyword-based features, can lead to statistically significant improvement in ad hoc document ranking especially on harder queries, i.e., those queries that are not handled efficiently by the baseline ($NDCG < 0.5$).

2 EXPERIMENTAL METHODOLOGY

Dataset: We ran our experiments on the LETOR 4.0 dataset for Learning to Rank. The document collection used in LETOR 4.0 is the Gov2 web page collection, which includes 25M pages. For the queries, we use the query set from the Million Query track TREC 2008, often referrred to as MQ2008, which consists of 800 queries. Given that our features are based on both word vectors as well as entities, we employed TAGME to semantically annotate documents and queries. In addition, we have used the pre-trained Word2Vec vectors based on the Google News corpus (3 billion words) with a vector size of 300[1] in our word embedding features and pre-trained Doc2Vec vectors based on English Wikipedia DBOW also with a vector size of 300[2] in our document embedding features.
Base Retrieval Model: As mentioned in Table 1, we use the 46 features included in LETOR 4.0 and explained in [10]. These features are used to train different learn to rank models to serve as baselines.
Ranking Models: We used four ranking models, which include two *listwise* models, namely AdaRank and ListNet, as well as two *pairwise* models, including RankerNet and RankBoost. All methods are trained and tested using a ten-fold cross-validation strategy.
Evaluation Metric: All methods are evaluated based on NDCG@20 as suggested in [6]. Statistical significance tests are performed by the Fisher Randomization (permutation) test.

Conference'17, Washington, DC, USA

DOI: 10.1145/nnnnnnn.nnnnnnn

[1] http://goo.gl/GxrfRm
[2] http://goo.gl/83BmeK

Table 1: Baseline and Embedding Features.

Feature Type		Feature Description	# of Features
Word Embedding	Word	(Average/Max) Cosine similarity between pairs of vector of words in document body/title/keyword/description and vector of words that appear in the query	8
	Entity	(Average/Max) Cosine similarity between pairs of vector of words in the abstracts of entities that appear in document body/title/keyword/description and vectors of words in the abstracts of entities in the query	8
Document Embedding	Word	Cosine similarity between the vector for document body/title/keyword/description and the vector for the query	4
	Entity	(Average/Max) Cosine similarity between the vector for entity abstracts in the document body/title/keyword/description and the vector for entity abstracts in the query	8
	Chunk	(Average/Max) Cosine similarity between vectors of chunks (non-overlapping windows of size 10/30/50) from document body and the vector for the query	6
Baseline		LETOR 4.0 ranking datasets features	46

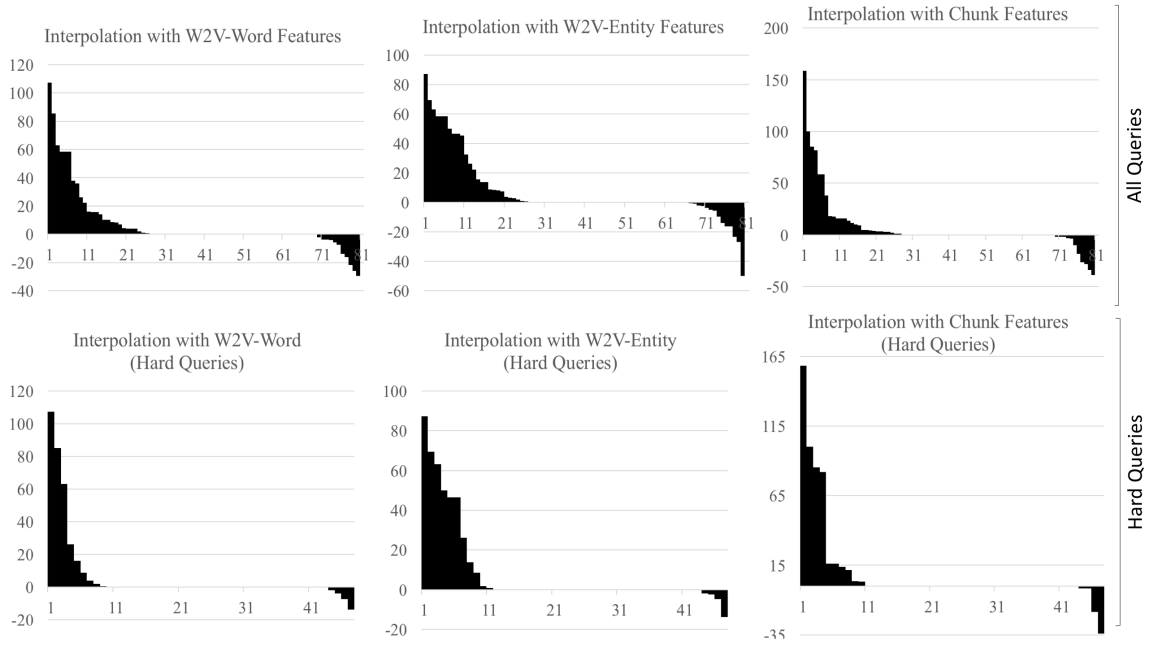

Figure 1: The X-axis lists all queries (top row) and hard queries with `NDCG<0.5` (bottom row), ordered by relative accuracy. The Y-axis is the relative accuracy of the baseline LETOR 4.0 features compared with the `interpolated` results using NDCG@20 using `AdaRank`. A positive value indicates an improvement and a negative value indicates a loss.

3 EVALUATION RESULTS

We first report the performance of the embedding features on all of the queries and then show their performance on harder queries.

3.1 All Queries

The results of our experiments are shown in Tables 2 and 3. Statistical significant improvements over the baseline are shown with ▲ while statistically significant lower performance compared to the baseline are specified using ▽. The results in these two tables under the 'All Queries' column show that regardless of the ranking model type, be it listwise or pairwise, embedding features alone do not perform well in the ranking task. In fact, embedding features when used in ListNet, RankerNet and RankBoost result in statistically significant lower performance compared to the baseline. However, when the embedding features are interpolated with the LETOR 4.0 features, in most cases ListNet, RankerNet and RankBoost show small yet statistically insignificant improvement over the baseline. There are only four cases of improvement in these three rankers that have resulted in statistically significant improvement over the baseline, which are all related to Word Embedding features, namely the interpolation of Word Embedding (Entity) in ListNet, interpolation of Word Embedding (Word) in RankerNet, interpolation of Word Embedding (Word) and interpolation of Word Embedding (Entity) in RankBoost. In contrast, AdaRank has shown to be receptive

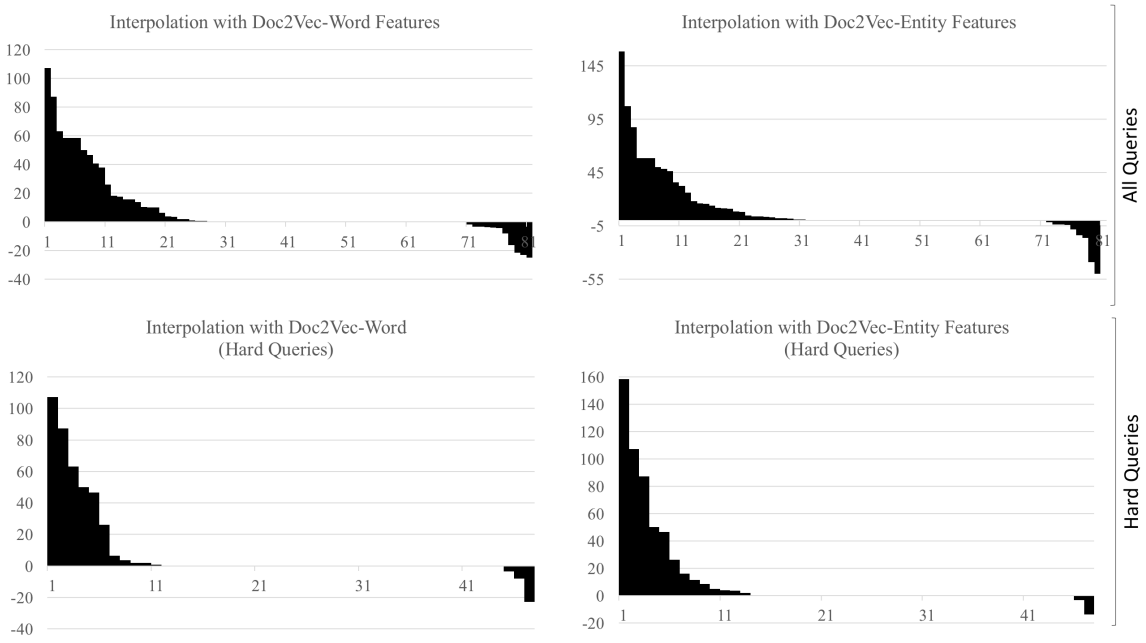

Figure 2: The X-axis lists all queries (top row) and hard queries with `NDCG<0.5` (bottom row), ordered by relative accuracy. The Y-axis is the relative accuracy of the baseline LETOR 4.0 features compared with the `interpolated` results using NDCG@20 using `AdaRank`. A positive value indicates an improvement and a negative value indicates a loss.

Table 2: Accuracy of learning to rank methods for `Listwise` rankers. Relative improvements over baseline are shown in parentheses. Triangles show statistical significance.

		Listwise			
		AdaRank		ListNet	
		All Queries	Hard Queries	All Queries	Hard Queries
Baseline		0.4215	0.1256	0.4564	0.1001
Word Embedding	Word (Embedding)	0.3876 (-8.03%)▽	0.1457 (15.93%)▲	0.3861 (-15.42%)▽	0.126 (25.88%)▲
	Word (Interpolation)	0.4578 (8.61%)▲	0.1529 (21.69%)▲	0.4672 (2.37%)	0.1222 (22.03%)▲
	Entity (Embedding)	0.3766 (-10.65%)▽	0.1669 (32.82%)▲	0.3845 (-15.77%)▽	0.1294 (29.27%)
	Entity (Interpolation)	0.4547 (7.9%)▲	0.1613 (28.35%)▲	0.4716 (3.33%)▲	0.1175 (17.42%)▲
Document Embedding	Word (Embedding)	0.402 (-4.61%)	0.1535 (22.15%)▲	0.4042 (-11.43%)▽	0.1364 (36.22%)▲
	Word (Interpolation)	0.4641 (10.13%)▲	0.1592 (26.72%)▲	0.4563 (-0.4%)	0.1085 (8.39%)▲
	Entity (Embedding)	0.3861 (-8.4%)	0.1584 (26.09%)	0.3924 (-14.03%)▽	0.1536 (53.43%)▲
	Entity (Interpolation)	0.4671 (10.84%)▲	0.1708 (35.92%)▲	0.4637 (1.58%)	0.1082 (8.05%)▲
	Chunk (Embedding)	0.4114 (-2.38%)	0.1719 (36.8%)▲	0.4186 (-8.69%)	0.1636 (63.42%)▲
	Chunk (Interpolation)	0.4564 (8.29%)▲	0.1673 (33.15%)▲	0.463 (1.43%)	0.1205 (20.32%)▲

of the interpolation of features. In all five interpolated embedding features, AdaRank shows statistically significant improvement.

We have also reported our findings with regards to the success/failure analysis of the queries in Figures 1 and 2 in the 'All Queries' row. The figures provide an analysis of the queries whose effectiveness are helped/hurt through the adoption of the interpolated embedding features. All the help/hurts were determined by comparing the relative difference percentage of NDCG@20 of the interpolated embedding features compared to the baseline and have been reported as percentages on the Y-axis. As seen in the figure, the number of queries that have been positively helped outnumber the number of queries that have been hurt, which is a positive performance indicator for the interpolated embedding features compared to the baseline.

3.2 Hard Queries

Now, in order to investigate whether the embedding features are more effective for easier queries or hard queries, we further report our findings specifically for the harder queries. We define hard queries to be those that have an NDCG@20 value lower than 0.5 by the baseline. This means that the baseline has a difficult time in effectively retrieving and ranking the best documents for these

Table 3: Accuracy of learning to rank methods for `Pairwise` rankers. Relative improvements over baseline are shown in parentheses. Triangles show statistical significance.

		Pairwise			
		RankerNet		RankBoost	
		All Queries	Hard Queries	All Queries	Hard Queries
Baseline		0.4663	0.1092	0.4617	0.1042
Word Embedding	Word (Embedding)	0.3971 (-14.83%)▽	0.1106 (1.28%)	0.406 (-12.06%)▽	0.1313 (25.95%)▲
	Word (Interpolation)	0.4806 (3.06%)▲	0.1298 (18.83%)▲	0.4764 (3.19%)▲	0.1296 (24.37%)▲
	Entity (Embedding)	0.3792 (-18.67%)▽	0.1197 (9.57%)	0.3864 (-16.3%)▽	0.147 (41.07%)▲
	Entity (Interpolation)	0.4665 (0.05%)	0.1272 (16.42%)▲	0.4743 (2.74%)▲	0.1198 (14.98%)
Document Embedding	Word (Embedding)	0.3918 (-15.98%)▽	0.1126 (3.06%)	0.4044 (-12.4%)▽	0.1277 (22.55%)
	Word (Interpolation)	0.4695 (0.69%)	0.1228 (12.42%)▲	0.4688 (1.54%)	0.1233 (18.26%)▲
	Entity (Embedding)	0.4023 (-13.71%)▽	0.127 (16.29%)	0.3922 (-14.9%)▽	0.131 (25.67%)▲
	Entity (Interpolation)	0.4715 (1.11%)	0.125 (14.48%)▲	0.4723 (2.31%)	0.1229 (17.87%)▲
	Chunk (Embedding)	0.3982 (-14.59%)▽	0.1302 (19.23%)▲	0.3966 (-11.09%)▽	0.129 (23.77%)▲
	Chunk (Interpolation)	0.4675 (0.26%)	0.1275 (16.71%)▲	0.4712 (2.06%)	0.1219 (16.99%)▲

queries. The results for the hard queries are reported in the 'Hard Queries' columns of Tables 2 and 3. The important observation based on the results reported in these two tables is that the embedding features have been quite effective on the hard queries by reporting statistically significant improvements over the baseline. In the AdaRank ranker, all the features including both embedding features (except Document Embedding (Entity)) as well as the interpolated embedding features have shown to improve the baseline with statistical significance. ListNet and RankBoost also show a similar statistically significant improvement over the baseline in hard queries. While RankerNet showed degraded performance using the embedding features and non-statistically significant improvement using the interpolated features on 'all queries', it showed statistically significant improvement on hard queries when interpolated embedding features were used.

One of the important observations on the hard queries was that the best performing feature sets for all of the rankers were related to the cases when embedding features were used alone. More specifically, the best performance on hard queries was obtained when Document Embedding (Chunk) features were used in AdaRank, ListNet and RankerNet and also when Word Embedding (Entity) features were used in RankBoost. Overall, both embedding features as well as the interpolated embedding features have been quite effective in improving ranking performance over hard queries. From a ranker perspective, listwise rankers observe the most statistically significant improvement over the baseline on the hard queries, which is inline with our observation over all queries.

Similar to 'all queries', we have also reported the success/failure analysis of the hard queries in Figures 1 and 2 in the 'Hard Queries' rows. It can be seen in these figures that the number of helped hard queries is much larger than the number of hurt hard queries. In addition, when comparing each chart in the bottom row (hard queries) with its corresponding chart on the top row (all queries), it can be seen that the number of hurt queries is reduced.

4 CONCLUDING REMARKS

The findings of this empirical study can be summarized as follows:

(1) Embedding features and their interpolation with LETOR 4.0 features show statistically significant improvement over the baseline on hard queries and can hence be considered strong features for handling queries that are difficult to handle by the baseline;
(2) As also reported in [2], while embedding and semantic features do not perform strongly when used independently, they result in increased performance when interpolated with keyword-based features, most specifically when used on harder queries. This could point to the fact that embedding features cover aspects of queries and documents that might not be covered by keyword-based features and hence result in increased ranking performance.
(3) Listwise rankers report the most consistent statistically significant improvement over all types of embedding and interpolated embedding features on hard queries, among which the Document Embedding (Chunk) features report the highest NDCG@20.
(4) Pairwise rankers showed to be the hardest to improve in combination with embedding and interpolated features; however, the best NDCG@20 result over all queries were obtained by pairwise rankers on interpolated Word Embedding (Word) features.

REFERENCES

[1] Michael Bendersky, W Bruce Croft, and Yanlei Diao. Quality-biased ranking of web documents. In *WSDM 2011*.
[2] Faezeh Ensan and Ebrahim Bagheri. Document Retrieval Model Through Semantic Linking. In *WSDM 2017*.
[3] Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity Hierarchy Embedding.. In *ACL (1)*. 1292–1300.
[4] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML 2014*.
[5] Craig Macdonald, B Taner Dinçer, and Iadh Ounis. Transferring Learning To Rank Models for Web Search. In *ICTIR 2015*.
[6] Craig Macdonald, Rodrygo L.T. Santos, and Iadh Ounis. On the Usefulness of Query Features for Learning to Rank. In *CIKM 2012*.
[7] Donald Metzler and W Bruce Croft. A Markov random field model for term dependencies. In *SIGIR 2005*.
[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS 2013*.
[9] Bhaskar Mitra and Nick Craswell. Neural Text Embeddings for Information Retrieval. In *WSDM 2017*.
[10] Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597* (2013).
[11] Chenyan Xiong, Russell Power, and Jamie Callan. Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding. In *WWW 2017*.