Context-Aware Term Weighting For First Stage Passage Retrieval

Zhuyun Dai
Carnegie Mellon University
zhuyund@cs.cmu.edu

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

ABSTRACT

Term frequency is a common method for identifying the importance of a term in a document. But term frequency ignores how a term interacts with its text context, which is key to estimating document-specific term weights. This paper proposes a Deep Contextualized Term Weighting framework (DeepCT) that maps the contextualized term representations from BERT to into context-aware term weights for passage retrieval. The new, deep term weights can be stored in an ordinary inverted index for efficient retrieval. Experiments on two datasets demonstrate that DeepCT greatly improves the accuracy of first-stage passage retrieval algorithms.

KEYWORDS

Deep Retrieval, Document Understanding, Term Weighting

ACM Reference Format:
Zhuyun Dai and Jamie Callan. 2020. Context-Aware Term Weighting For First Stage Passage Retrieval . In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397271.3401204

1 INTRODUCTION

State-of-the-art search engines use ranking pipelines in which an efficient *first stage* uses a query to fetch an initial set of documents, and one or more *re-ranking* algorithms to improve and prune the ranking. Typically the first stage ranker is bag-of-words retrieval model that use term frequency (*tf*) to determine the document-specific importance of terms. However, *tf* does not necessarily indicate whether a term essential to the meaning of the document, especially when the frequency distribution is flat, e.g., passages. In essence, *tf* ignores the interactions between a term and its *text context*, which is key to estimating document-specific term weights.

This paper seeks to improve term importance estimation in first-stage retrieval models by using deep language models like BERT [3] to capture a term's contextual features. We present the Deep Contextualized Term Weighting framework (DeepCT). DeepCT learns a contextualized term representation model based on BERT, and a mapping function from representations to term weights. The transformer encoder of BERT allows DeepCT to capture semantic and syntactic features from a term's linguistic context, helping DeepCT to identify semantically important terms from the text. *DeepCT generates deep, context-aware document-specific term weights that can replace the standard tf.*

This paper also present a novel approach that runs DeepCT at offline index time, making it possible to use it in first-stage retrieval where efficiency is crucial. Our approach applies DeepCT over each passage in the corpus, and stores the context-aware term weights in an ordinary inverted index to replace *tf*. The index can be searched efficiently using common bag-of-words retrieval models such as BM25 or statistical query likelihood models.

Experiments demonstrate that DeepCT significantly improves the accuracy of first-stage retrieval. More accurate first-stage document rankings also provide better candidates for downstream reranking, and improves end-to-end accuracy and/or efficiency. Analysis shows that DeepCT's main advantage is the ability to differentiate between key terms and other frequent but non-central terms. Code and data are made public available [1].

2 RELATED WORK

Document Term Weighting. Most first-stage retrieval models such as BM25 and query likelihood use term frequencies (*tf*) to term importance in a document. A popular alternative to *tf* are graph-based methods, e.g., TextRank [6]. A few recent work investigated using word embeddings [5] for document term weighting, but most of them only learn a global *idf*-like term weight because the word embeddings are context-independent. Our work aims to learn *tf*-like term weights that are context-specific.

Neural Approaches for First Stage Ranking. Most neural ranking models are cost-prohibitive to be used in the first stage [1, 2, 10]. Recent research addresses this efficiency problem in two ways. One way is to learn latent embedding representations of queries and documents [13]. However, fix-dimension representations introduce the specificity vs. exhaustiveness trade-off [14]. Another approach modifies the bag-of-words document representations using neural network. Mitra et al. [8] uses neural rankers to generate term-document scores, but it is time-consuming when applied at a large-scale. Nogueira et al. [11] proposed to generate queries from documents using neural machine translation and index queries as document expansion terms [11, 12]. Our work are orthogonal to [11, 12] – their approaches *add* terms to documents, while our method *weights* existing terms; future work may consider combining the two approaches.

3 DeepCT

DeepCT Framework DeepCT leverages the transformer encoder of BERT to extract a word's contextual features. In the transformer, a term gradually absorbs contextual information based on its attention to every other term in the same text. At the last layer, the transformer generates a contextualized term embedding for every

[1] https://github.com/AdeDZY/DeepCT

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07.
https://doi.org/10.1145/3397271.3401204

term in the input text, which can be viewed as a feature vector that characterizes the term's syntactic and semantic role in a given context.

DeepCT then linearly combines a term's contextualized embedding into a term importance score:

$$\hat{y}_{t,d} = \vec{w} T_{t,d} + b \tag{1}$$

where $T_{t,d}$ is term t's contextualized embedding in text d; and, $\vec{w}$ and b are the linear combination weights and bias.

Train DeepCT DeepCT is trained end-to-end using a per-token regression task. Given the ground truth term weight for every word in text d, denoted as $\{y_{1,d}, \ldots, y_{n,d}\}$, DeepCT aims to minimize the mean square error (MSE) between the predicted weights $\hat{y}$ and the ground truth weights y:

$$loss_{MSE} = \sum_{d} \sum_{t} (y_{t,d} - \hat{y}_{t,d})^2. \tag{2}$$

It is an open question how to obtain the ground truth term weights y. As search queries can reflect the key idea of a document, we propose *query term recall* as an estimation of the ground truth term weights:

$$y_{t,d} = QTR(t,d) = \frac{|Q_{d,t}|}{|Q_d|}. \tag{3}$$

Q_d is the set of queries for which passage d is judged relevant. $Q_{d,t}$ is the subset of Q_d that contains term t. In other words, $QTR(t,d)$ is the percentage of d's queries that mention term t.

Index with DeepCT The queries are only used to generate training labels. During testing, the model is *query-independent* – the prediction is solely based on the document content. This allows estimated term weights to be calculated and stored during offline indexing.

We run the trained DeepCT model over all passages in the collection. Most predictions fall into 0-1 (because the ground truth weights are in 0-1, as shown in Eq.3). The predicted weights are then scaled into an integer that can be used with existing retrieval models. We call the scaled weight tf_{DeepCT} to convey that it is an alternate way of representing the importance of term t in d:

$$tf_{\text{DeepCT}}(t,d) = round(\hat{y}_{t,d} * N). \tag{4}$$

N scales the predicted weights into a integer range. We use $N = 100$ in this work as two digits of precision is sufficient for this task. Terms with negative weights are discarded.

tf_{DeepCT} is used to *replace* the original tf in the inverted index. The new index, DeepCT-Index, can be searched by mainstream bag-of-words retrieval model like BM25 or query likelihood model. The context-aware term weight tf_{DeepCT} is expect to bias the retrieval models to central terms in the pasessag, preventing off-topic passages being retrieved.

Efficiency The main difference between DeepCT-Index and a typical inverted index is that the term weight is based on tf_{DeepCT} instead of tf. This calculation is done offline. No new posting lists are created, thus the query latency does not become longer. To the contrary, a side-effect of Eq 4 is that tf_{DeepCT} of some terms becomes negative, which may be viewed as a form of index pruning. We leave that aspect of this work for future investigation.

4 EXPERIMENTAL METHODOLOGY

Datasets used MS MARCO [9] and TREC-CAR [4]. **MS MARCO** is a passage retrieval dataset with 8.8M passages [9]. The training set contains approximately 0.5M pairs of queries and relevant passages. The development (dev) set contains 6,980 queries and their relevance labels. The test set contains 6,900 queries, but the relevance labels are hidden by Microsoft. Therefore, *the dev set is our main evaluation set.* In a few experiments, we also evaluated on the test set by submitting our rankings to the MS MARCO competition. **TREC-CAR** [4] consists of 29.7M English Wikipedia passages. Following prior work [10, 11], we use the automatic relevance judgments. The training set have 3.3M query-passage pairs. The test set contains 1,860 queries.

First-Stage Retrieval Baselines. We compare DeepCT term weights with three popular term weighting methods used in first-stage retrieval.

- *tf* uses standard term frequency weights, e.g., as used by BM25.
- TextRank [6] is a widely-used graph-based term weighting approach. We use the open source PyTextRank implementation[2]. Term weights from TextRank are in the range (0, 1); we scale them to integers as described in Eq. 4 for indexing.
- Doc2Query [11] is a supervised baseline. It trains a neural sequence-to-sequence model to generate potential queries from passages, and indexes the queries as document expansion terms. We use the Doc2Query MS MARCO index released by the authors. No such index is available for TREC-CAR, so we use published values for that dataset [11].

We used the Anserini toolkit to index documents using the above three term weights as well as the proposed DeepCT term weights. First-stage ranking was done by two popular retrieval models: BM25 and query likelihood with Jelinek-Mercer smoothing (QL). We fine-tuned BM25 parameters k_1 and b, and QL smoothing factor λ through a parameter sweep on 500 queries from the training set.

The transformer of DeepCT was initialized with pre-trained BERT parameters. For MS MARCO, we used the official pre-trained BERT (uncased, base model) [3]. For TREC-CAR, we follow Nogueira and Cho [10] and used a pre-trained BERT model released by the authors. DeepCT was trained for 3 epochs on the training split of our datasets, using a learning rate of $2e^{-5}$ and a max input text length of 128 tokens.

Evaluation used MRR@10, the official MS MARCO evaluation metric. For TREC-CAR, we also report MAP at depth 1,000 following the evaluation methodology used in previous work [10, 11].

5 EXPERIMENTAL RESULTS

Three experiments investigate DeepCT's first-stage retrieval accuracy, its impacts on down-stream rerankers, and why DeepCT term weights are effective.

5.1 First-Stage Retrieval using DeepCT

The first experiment examines whether DeepCT improves first-stage retrieval accuracy over baseline retrieval methods. It also compares DeepCT to several supervised rerankers.

[2] https://github.com/DerwenAI/pytextrank

Table 1: Ranking accuracy of BM25 and QL using indexes built with three baselines and three DeepCT methods. Win/Tie/Loss are the number of queries improved, unchanged, or hurt, compared to tf index on MRR@10. * and † indicates statistically significant improvements over tf index and Doc2Query.

Index	MS MARCO dev				TREC-CAR					
	BM25		QL		BM25			QL		
	MRR@10	W/T/L	MRR@10	W/T/L	MRR@10	MAP	W/T/L	MRR@10	MAP	W/T/L
tf	0.191	-/-/-	0.189	-/-/-	0.233	0.174	-/-/-	0.211	0.162	-/-/-
TextRank	0.130	662/4556/1762	0.134	702/4551/1727	0.160	0.120	167/1252/441	0.157	0.118	166/1327/367
Doc2Query	0.221*	1523/4431/1026	0.224*	1603/4420/957	-	0.178	-/-/-	-	-	-/-/-
DeepCT	**0.243***†	2022/3861/1097	**0.230***	1843/4027/1110	**0.332***	**0.246***	615/1035/210	**0.330***	**0.247***	645/1071/144

Table 2: Retrieval accuracy of BM25 with DeepCT compared with several supervised rerankers on the MS MARCO dataset using official evaluation on dev and hidden test set.[3]

Ranking Method		dev MRR@10		test MRR@10	
Single-Stage	Official BM25	0.167	-30%	0.165	-31%
	DeepCT BM25 (*this work*)	*0.243*	–	*0.239*	–
Multi-Stage	Feature-based LeToR	0.195	-20%	0.191	-20%
	K-NRM [15]	0.218	-10%	0.198	-17%
	Duet V2 [7]	0.243	+0%	0.245	+2%
	Conv-KNRM [2]	0.247	+2%	0.247	+3%
	BERT ReRanker [10]	0.365	+50%	0.359	+50%

Table 3: Reranking accuracy of two rerankers applied to passages retrieved using BM25 from DeepCT index and the tf index. Dataset: MS MARCO dev.

	Recall		Conv-KNRM reranker MRR@10		BERT reranker MRR@10	
Depth	tf	DeepCT	tf	DeepCT	tf	DeepCT
10	40%	49%	0.234	0.270	0.279	0.320
20	49%	58%	0.244	0.277	0.309	0.343
100	68%	76%	0.256	0.274	0.349	0.368
200	75%	82%	0.256	0.269	0.358	0.370
1000	86%	91%	0.256 [1]	0.264	0.371 [1]	0.376

[1] The values are not exactly the same as in Table 2 due to differences in the initial rankings generated from our BM25 and the official BM25 from MS MSARCO.

Comparison to First-stage Retrieval Baselines. Table 1 shows the first-stage retrieval accuracy of BM25 and QL using indexes generated by four methods. The TextRank index failed to beat the common tf index. The Doc2Query index was effective for MS MARCO, but only marginally better for TREC-CAR. On the other hand, DeepCT outperformed the baselines by large margins. It improved BM25 by 27% on MS MARCO and 46% on TREC-CAR. It produced similar gains for QL, showing that DeepCT is useful to different retrieval models. DeepCT-Index also surpassed Doc2Query by large margins, which also used deep neural networks and supervised training.

It is uncommon in prior research for a non-*tf* term weighting method to generate such substantially better rankings. These results show that *tf* is no longer sufficient, and that better term importance signals can be generated with deep document understanding.

Comparision to Supervised Rerankers. We further compared the single-stage DeepCT retrieval to several multi-stage reranking pipelines. Table 2 shows results from the MS MARCO leaderboard[4]. It lists representative reranking approaches for feature-based learning-to-rank, previous state-of-the-art neural rerankers (*non-ensemble versions*), and BERT-based rerankers. All rerankers used the top 1,000 passages from BM25.

A single-stage BM25 retrieval from DeepCT-Index was better than several reranking pipelines. It is more accurate than feature-based LeToR, a widely used reranking approach in modern search engines. It is also more accurate than a popular neural reranking model K-NRM [15]. Compared to Duet V2 (the official reranking baseline) and Conv-KNRM [2], DeepCT achieves similar accuracy while being more efficient as it does not need the reranking stage. The results demonstrate that it is possible to move some of the complex ranking process to offline analysis, building deep yet simple text representations that can be retrieved very efficiently.

Finally, strong neural rerankers like the BERT reranker were much more effective than DeepCT BM25. The next section studies whether the two approaches can be used together.

5.2 Combine DeepCT with Rerankers

This experiment examines whether a first-stage ranking produced by DeepCT BM25 can be used together with later-stage rerankers to improve end-to-end performance. Table 3 reports the performance of two rerankers applied to candidate passages retrieved from the standard *tf* and the DeepCT index. The two rerankers were Conv-KNRM [2], which has medium accuracy, and BERT ReRanker [10], which has high accuracy. We tested various reranking depths. Reranking at a shallower depth has higher efficiency but may miss more relevant passages.

The recall values show the percentage of relevant passages in the reranking passage set. DeepCT had higher recall at all depths, meaning a ranking from DeepCT provided more relevant passages to a reranker. Both rerankers consistently achieved higher MRR@10 by using DeepCT compared to using *tf*. For Conv-KNRM, the best

[3] Statistical significance is unknown because the MS MARCO website publishes only summary results.

[4] http://www.msmarco.org/leaders.aspx

[4] We did not run this experiment on MS MARCO test set because test set results can only be evaluated by submitting to the MS MARCO competition. The organizers discourage too many submission from the same group to avoid "P-hacking".

Table 4: Visualization of DeepCT term weights. Red shades reflect the normalized term weights. Query terms are bold.

	0 10% 20% 30% 40% >50%
Query	**do atoms make up dna**
On-Topic	**DNA** only has 5 different **atoms** - carbon, hydrogen, oxygen, nitrogen and phosphorous. According to one estimation, there are about 204 billion **atoms** in each **DNA**.
Off-Topic	Genomics in Theory and Practice. What is Genomics. Genomics is a study of the genomes of organisms. It main task is to determine the entire sequence of **DNA** or the composition of the **atoms** that make up the **DNA** and the chemical bonds between the **DNA atoms**.

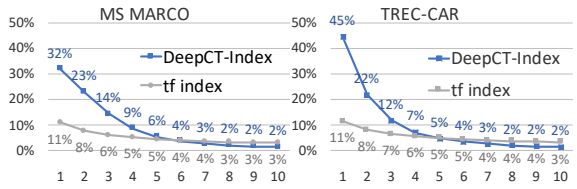

Figure 1: Term weight distribution of the top-10 terms in passages with highest weights. The X-axis shows the term's rank ordered by weight. The Y-axis shows the average term weight normalized by total passage term weight.

MRR@10 improved from 0.256 to 0.278, and the required reranking depth decreased from 100 to 50. For BERT ReRanker, DeepCT enabled it to achieve similar accuracy using much fewer passages. Reranking the top 100-200 passages from DeepCT produced similar MRR@10 as reranking the top 1,000 passages from *tf* index, meaning that the reranker can be 5-10× more efficient.

In summary, DeepCT puts relevant passages at the top, so that downstream rerankers can achieve similar or higher accuracy with much smaller candidate sets, leading to lower computational cost in the retrieval pipeline.

5.3 Sources of Effectiveness

The last experiment aims to understand the sources of effectiveness of DeepCT-Index through several analyses.

Table 4 visualizes DeepCT weights. It shows that DeepCT is able to *emphasize central terms and suppress non-central terms*. Non-central terms are assigned with low weight even they are frequent. For example, in the off-topic passage, *"DNA"* has low DeepCT weight even though it is mentioned 3 times. On the other hand, *"Genomics"* has higher weight even though is is mentioned fewer times. This extent of independence from frequency signals is uncommon in previous term weighting approaches.

Figure 1 compares the term weight distribution of DeepCT-Index and `tf` index.The original *tf* distribution is flat. DeepCT-Index assigns high weights to a few central terms, resulting in a skewed term weight distribution. Such skewed distribution confirms our observations from the case study that DeepCT-Index aggressively emphasizes a few central terms and supresses the others.

6 CONCLUSION

Most first-stage rankers are efficient bag-of-words retrieval models that use term frequency signals. This paper presents DeepCT, a deep learning approach that better estimates term importance for first stage retrieval. DeepCT uses the transformer encoder of BERT to capture contextual features of words, and maps the features into document-specific term weights. Trained on a supervised per-token regression task, DeepCT is capable of producing context-aware term weights that reflect the essential meanings of the document. Importantly, DeepCT moves the neural document processing to the *offline* indexing time – the term weights can be stored in a typical inverted index and used with efficient retrieval models such as BM25.

Experimental results show that DeepCT improves the accuracy of popular first-stage retrieval algorithms by up to 40%. Running BM25 on DeepCT-Index can be as effective as several previous state-of-the-art rankers that need to run slow deep learning models at the query time. The higher-quality ranking enabled by DeepCT-Index improves the accuracy/efficiency tradeoff for later-stage re-rankers. Analysis shows that DeepCT is capable of finding the central words in a text even if they are mentioned only once. We view DeepCT as an encouraging step from "frequencies" to " meanings".

For several decades, first-stage retrieval models have relied on term frequency signals (*tf*). Results from this paper indicate that *tf* is no longer sufficient. With recent advances in deep learning and NLP, it is time to revisit the indexers and retrieval models, towards building new deep and efficient first stage rankers.

Acknowledgments. This work was supported by the National Science Foundation (NSF) grant IIS-1815528.

REFERENCES

[1] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. In *Proc. SIGIR*.
[2] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proc. WSDM*.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. ACL*.
[4] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. 2017. TREC complex answer retrieval overview. In *Proc. TREC*.
[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proc. CIKM*.
[6] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proc. EMNLP*.
[7] Bhaskar Mitra and Nick Craswell. 2019. An Updated Duet Model for Passage Re-ranking. *arXiv preprint arXiv:1903.07666* (2019).
[8] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. 2019. Incorporating Query Term Independence Assumption for Efficient Retrieval and Ranking using Deep Neural Networks. *arXiv preprint arXiv:1907.03693* (2019).
[9] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268* (2016).
[10] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).
[11] Rodrigo Nogueira, Kyunghyun Cho, Yang Wei, Lin Jimmy, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv:1904.08375* (2019).
[12] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docTTTTTquery. (2019).
[13] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.
[14] Gerard Salton and Michael McGill. 1984. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company.
[15] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proc. SIGIR*.