CEDR: Contextualized Embeddings for Document Ranking

Sean MacAvaney
IRLab, Georgetown University
sean@ir.cs.georgetown.edu

Andrew Yates
Max Planck Institute for Informatics
ayates@mpi-inf.mpg.de

Arman Cohan
Allen Institute for Artificial Intelligence
armanc@allenai.org

Nazli Goharian
IRLab, Georgetown University
nazli@ir.cs.georgetown.edu

ABSTRACT

Although considerable attention has been given to neural ranking architectures recently, far less attention has been paid to the term representations that are used as input to these models. In this work, we investigate how two pretrained contextualized language models (ELMo and BERT) can be utilized for ad-hoc document ranking. Through experiments on Trec benchmarks, we find that several existing neural ranking architectures can benefit from the additional context provided by contextualized language models. Furthermore, we propose a joint approach that incorporates BERT's classification vector into existing neural models and show that it outperforms state-of-the-art ad-hoc ranking baselines. We call this joint approach CEDR (Contextualized Embeddings for Document Ranking). We also address practical challenges in using these models for ranking, including the maximum input length imposed by BERT and runtime performance impacts of contextualized language models.

ACM Reference Format:
Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized Embeddings for Document Ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331317

1 INTRODUCTION

Recently, there has been much work designing ranking architectures to effectively score query-document pairs, with encouraging results [5, 6, 20]. Meanwhile, pretrained contextualized language models (such as ELMo [16] and BERT [4]) have shown great promise on various natural language processing tasks [4, 16]. These models work by pre-training LSTM-based or transformer-based [19] language models on a large corpus, and then by performing minimal task fine-tuning (akin to ImageNet [3, 23]).

Prior work has suggested that contextual information can be valuable when ranking. ConvKNRM [1], a recent neural ranking model, uses a convolutional neural network atop word representations, allowing the model to learn representations aware of context in *local* proximity. In a similar vein, McDonald et al. [12] proposes an approach that learns a recurrent neural network for term representations, thus being able to capture context from the entire text [12]. These approaches are inherently limited by the variability found in the training data. Since obtaining massive amounts of high-quality relevance information can be difficult [24], we hypothesize that *pretrained* contextualized term representations will improve ad-hoc document ranking performance.

We propose incorporating contextualized language models into existing neural ranking architectures by using multiple similarity matrices – one for each layer of the language model. We find that, at the expense of computation costs, this improves ranking performance considerably, achieving state-of-the-art performance on the Robust 2004 and WebTrack 2012–2014 datasets. We also show that combining each model with BERT's classification mechanism can further improve ranking performance. We call this approach CEDR (Contextualzed Embeddings for Document Ranking). Finally, we show that the computation costs of contextualized language models can be dampened by only partially computing the contextualized language model representations. Although others have successfully used BERT for *passage* ranking [14] and question answering [22], these approaches only make use of BERT's sentence classification mechanism. In contrast, we use BERT's term representations, and show that they can be effectively combined with existing neural ranking architectures.

In summary, our contributions are as follows:

- We are the first to demonstrate that contextualized word representations can be successfully incorporated into existing neural architectures (PACRR [6], KNRM [20], and DRMM [5]), allowing them to leverage contextual information to improve ad-hoc document ranking.
- We present a new joint model that combines BERT's *classification* vector with existing neural ranking architectures (using BERT's *token* vectors) to get the benefits from both approaches.
- We demonstrate an approach for addressing the performance impact of computing contextualized language models by only partially computing the language model representations.
- Our code is available for replication and future work.[1]

2 METHODOLOGY

2.1 Notation

In ad-hoc ranking, documents are ranked for a given query according to a relevance estimate. Let Q be a query consisting of query terms $\{q_1, q_2, ..., q_{|Q|}\}$, and let D be a document consisting of terms $\{d_1, d_2, ..., d_{|D|}\}$. Let $ranker(Q, D) \in \mathbb{R}$ be a function that returns

[1] https://github.com/Georgetown-IR-Lab/cedr

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331317

a real-valued relevance estimate for the document to the query. Neural relevance ranking architectures generally use a similarity matrix as input $\mathbf{S} \in \mathbb{R}^{|Q|\times|D|}$, where each cell represents a similarity score between the query and document: $\mathbf{S}_{i,j} = sim(q_i, d_j)$. These similarity values are usually the cosine similarity score between the word vectors of each term in the query and document.

2.2 Contextualized similarity tensors

Pretrained contextual language representations (such as those from ELMo [16] and BERT [4]) are context sensitive; in contrast to more conventional pretrained word vectors (e.g., GloVe [15]) that generate a single word representation for each word in the vocabulary, these models generate a representation of each word based on its context in the sentence. For example, the contextualized representation of word *bank* would be different in *bank deposit* and *river bank*, while a pretrained word embedding model would always result in the same representation for this term. Given that these representations capture contextual information in the language, we investigate how these models can also benefit general neural ranking models.

Although contextualized language models vary in particular architectures, they typically consist of multiple stacked layers of representations (e.g., recurrent or transformer outputs). The intuition is that the deeper the layer, the more context is incorporated. To allow neural ranking models to learn which levels are most important, we choose to incorporate the output of all layers into the model, resulting in a three-dimensional similarity representation. Thus, we expand the similarity representation (conditioned on the query and document context) to $\mathbf{S}_{\mathbf{Q},\mathbf{D}} \in \mathbb{R}^{L\times|Q|\times|D|}$ where L is the number of layers in the model, akin to the channel in image processing. Let $context_{\mathbf{Q},\mathbf{D}}(t,l) \in \mathbb{R}^D$ be the contextualized representation for token t in layer l, given the context of Q and D. Given these definitions, let the contextualized representation be:

$$\mathbf{S}_{\mathbf{Q},\mathbf{D}}[l,q,d] = cos(context_{\mathbf{Q},\mathbf{D}}(q,l), context_{\mathbf{Q},\mathbf{D}}(d,l)) \quad (1)$$

for each query term $q \in Q$, document term $d \in D$, and layer $l \in [1..L]$. Note that when q and d are identical, they will likely not receive a similarity score of 1, as their representation depends on the surrounding context of the query and document. The layer dimension can be easily incorporated into existing neural models. For instance, soft n-gram based models, like PACRR, can perform convolutions with multiple input channels, and counting-based methods (like KNRM and DRMM) can count each channel individually.

2.3 Joint BERT approach

Unlike ELMo, the BERT model encodes multiple text segments simultaneously, allowing it to make judgments about text pairs. It accomplishes this by encoding two meta-tokens (`[SEP]` and `[CLS]`) and using text segment embeddings (*Segment A* and *Segment B*). The `[SEP]` token separates the tokens of each segment, and the `[CLS]` token is used for making judgments about the text pairs. During training, `[CLS]` is used for predicting whether two sentences are sequential – that is, whether *Segment A* immediately precedes *Segment B* in the original text. The representation of this token can be fine-tuned for other tasks involving multiple text segments, including natural language entailment and question answering [22].

We explore incorporating the `[CLS]` token's representation into existing neural ranking models as a joint approach. This allows neural rankers to benefit from deep semantic information from BERT in addition to individual contextualized token matches.

Incorporating the `[CLS]` token into existing ranking models is straightforward. First, the given ranking model produces relevance scores (e.g., n-gram or kernel scores) for each query term based on the similarity matrices. Then, for models using dense combination (e.g., PACRR, KNRM), we propose concatenating the `[CLS]` vector to the model's signals. For models that sum query term scores (e.g., DRMM), we include the `[CLS]` vector in the dense calculation of each term score (i.e., during combination of bin scores).

We hypothesize that this approach will work because the BERT classification mechanism and existing rankers have different strengths. The BERT classification benefits from deep semantic understanding based on next-sentence prediction, whereas ranking architectures traditionally assume query term repetition indicates higher relevance. In reality, both are likely important for relevance ranking.

3 EXPERIMENT

3.1 Experimental setup

Datasets. We evaluate our approaches using two datasets: Trec Robust 2004 and WebTrack 2012–14. For Robust, we use the five folds from [7] with three folds used for training, one fold for testing, and the previous fold for validation. For WebTrack, we test on 2012–14, training each year individually on all remaining years (including 2009–10), and validating on 2011. (For instance, when testing on WebTrack 2014, we train on 2009–10 and 2012–13, and validate on 2011.) Robust uses Trec discs 4 and 5[2], WebTrack 2009–12 use ClueWeb09b[3], and WebTrack 2013–14 uses ClueWeb12[4] as document collections. We evaluate the results using the nDCG@20 / P@20 metrics for Robust04 and nDCG@20 / ERR@20 for WebTrack.

Models. Rather than building new models, in this work we use existing model architectures to test the effectiveness of various input representations. We evaluate our methods on three neural relevance matching methods: PACRR [6], KNRM [20], and DRMM [5]. Relevance matching models have generally shown to be more effective than semantic matching models, while not requiring massive amounts of behavioral data (e.g., query logs). For PACRR, we increase $k_{max} = 30$ to allow for more term matches and better back-propagation to the language model.

Contextualized language models. We use the pretrained ELMo (Original, 5.5B) and BERT (BERT-Base, Uncased) language models in our experiments. For ELMo, the query and document are encoded separately. Since BERT enables encoding multiple texts at the same time using *Segment A* and *Segment B* embeddings, we encode the query (*Segment A*) and document (*Segment B*) simultaneously. Because the pretrained BERT model is limited to 512 tokens, longer documents are split such that document segments are split as evenly as possible, while not exceeding the limit when combined with the query and control tokens. (Note that the query is always included in full.) BERT allows for simple classification fine-tuning, so we also experiment with a variant that is first fine-tuned on the

[2] 520k documents; https://trec.nist.gov/data_disks.html
[3] 50M web pages, https://lemurproject.org/clueweb09/
[4] 733M web pages, https://lemurproject.org/clueweb12/

same data using the Vanilla BERT classifier (see baseline below), and further fine-tuned when training the ranker itself.

Training and optimization. We train all models using pairwise hinge loss [2]. Positive and negative training documents are selected from the query relevance judgments (positive documents limited to only those meeting the re-ranking cutoff threshold k using BM25, others considered negative). We train each model for 100 epochs, each with 32 batches of 16 training pairs. Gradient accumulation is employed when the batch size of 16 is too large to fit on a single GPU. We re-rank to top k BM25 results for validation, and use P@20 on Robust and nDCG@20 on WebTrack to select the best-performing model. We different re-ranking functions and thresholds at test time for each dataset: BM25 with $k = 150$ for Robust04, and QL with $k = 100$ for WebTrack. The re-ranking setting is a better evaluation setting than ranking all qrels, as demonstrated by major search engines using a pipeline approach [18]. All models are trained using Adam [8] with a learning rate of 0.001 while BERT layers are trained at a rate of 2e-5.[5] Following prior work [6], documents are truncated to 800 tokens.

Baselines. We compare contextualized language model performance to the following strong baselines:

- BM25 and SDM [13], as implemented by Anserini [21]. Fine-tuning is conducted on the test set, representing the maximum performance of the model when using static parameters over each dataset.[6] We do not report SDM performance on WebTrack due to its high cost of retrieval on the large ClueWeb collections.
- Vanilla BERT ranker. We fine-tune a pretrained BERT model (`BERT-Base, Uncased`) with a linear combination layer stacked atop the classifier `[CLS]` token. This network is optimized the same way our models are, using pairwise cross-entropy loss and the Adam optimizer. We use the approach described above to handle documents longer than the capacity of the network, and average the `[CLS]` vectors from each split.
- `TREC-best`: We also compare to the top-performing topic TREC run for each track in terms of nDCG@20. We use `uogTrA44xu` for WT12 ([10], a learning-to-rank based run), `clustmrfaf` for WT13 ([17], clustering-based), `UDInfoWebAX` for WT14 ([11], entity expansion), and `pircRB04t3` for Robust04 ([9], query expansion using Google search results).[7]
- ConvKNRM [1], our implementation with the same training pipeline as the evaluation models.
- Each evaluation model when using GloVe [15] vectors.[8]

3.2 Results & analysis

Table 1 shows the ranking performance using our approach. We first note that the Vanilla BERT method significantly outperforms the tuned BM25 [V] and ConvKNRM [C] baselines on its own. This is encouraging, and shows the ranking power of the Vanilla BERT model. When using contextualized language term representations without tuning, PACRR and DRMM performance is comparable to that of GloVe [G], while KNRM sees a modest boost. This might be due to KNRM's ability to train its matching kernels, tuning to specific similarity ranges produced by the models. (In contrast, DRMM uses fixed buckets, and PACRR uses maximum convolutional filter strength, both of which are less adaptable to new similarity score ranges.) When fine-tuning BERT, all three models see a significant boost in performance compared to the GloVe-trained version. PACRR and KNRM see comparable or higher performance than the Vanilla BERT model. This indicates that fine-tuning contextualized language models for ranking is important. This boost is further enhanced when using the CEDR (joint) approach, with the CEDR models always outperforming Vanilla BERT [V], and nearly always significantly outperforming the non-CEDR versions [N]. This suggests that term counting methods (such as KNRM and DRMM) are complementary to BERT's classification mechanism. Similar trends for both Robust04 and WebTrack 2012–14 indicate that our approach is generally applicable to ad-hoc document retrieval tasks.

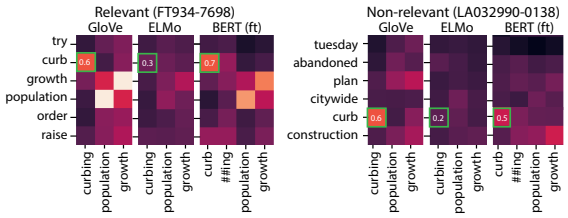

Figure 1: Example similarity matrix excerpts from GloVe, ELMo, and BERT for relevant and non-relevant document for Robust query 435. Lighter values have higher similarity.

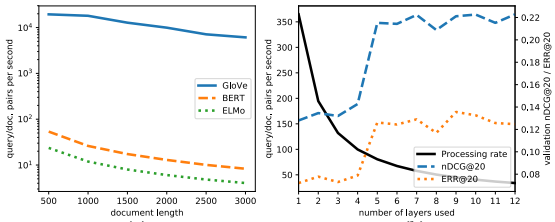

Figure 2: (a) Processing rates by document length for GloVe, ELMo, and BERT using PACRR. (b) Processing rate and dev performance of PACRR when using a subset of BERT layers.

To gain a better understanding of how the contextual language model helps enhance the input representation, we plot example similarity matrices based on GloVe word embeddings, ELMo representations (layer 2), and fine-tuned BERT representations (layer 5). In these examples, two senses of the word *curb* (restrain, and edge of street) are encountered. The first is relevant to the query (it's discussing attempts to restrain population growth). The second is not (it discusses street construction). Both the ELMo and BERT models give a higher similarity score to the correct sense of the term for the query. This ability to distinguish different senses of terms is a strength of contextualized language models, and likely can explain some of the performance gains of the non-joint models.

Although the contextualized language models yield ranking performance improvements, they come with a considerable cost at inference time—a practical issue ignored in previous ranking work [14, 21]. To demonstrate this, in Figure 2(a) we plot the processing rate of GloVe, ELMo, and BERT.[9] Note that the processing

[5] Pilot experiments showed that a learning rate of 2e-5 was more effective on this task than the other recommended values of 5e-5 and 3e-5 by [4].
[6] k_1 in 0.1–4 (by 0.1), b in 0.1–1 (by 0.1), SDM weights in 0–1 (by 0.05).
[7] We acknowledge limitations of the TREC experimentation environment.
[8] `glove.42B.300d`, https://nlp.stanford.edu/projects/glove/
[9] Running time measured on single GeForce GTX 1080 Ti GPU, data in memory.

Table 1: Ranking performance on Robust04 and WebTrack 2012–14. Significant improvements to [B]M25, [C]onvKNRM, [V]anilla BERT, the model trained with [G]lOve embeddings, and the corresponding [N]on-CEDR system are indicated in brackets (paired t-test, $p < 0.05$).

		Robust04		WebTrack 2012–14	
Ranker	Input Representation	P@20	nDCG@20	nDCG@20	ERR@20
BM25	n/a	0.3123	0.4140	0.1970	0.1472
SDM [13]	n/a	0.3749	0.4353	-	-
TREC-Best	n/a	**0.4386**	**0.5030**	0.2855	**0.2530**
ConvKNRM	GloVe	0.3349	0.3806	[B] 0.2547	[B] 0.1833
Vanilla BERT	BERT (fine-tuned)	[BC] 0.4042	[BC] 0.4541	**[BC] 0.2895**	[BC] 0.2218
PACRR	GloVe	0.3535	[C] 0.4043	0.2101	0.1608
PACRR	ELMo	[C] 0.3554	[C] 0.4101	[BG] 0.2324	[BG] 0.1885
PACRR	BERT	[C] 0.3650	[C] 0.4200	0.2225	0.1817
PACRR	BERT (fine-tuned)	[BCVG] 0.4492	[BCVG] 0.5135	[BCG] 0.3080	[BCG] 0.2334
CEDR-PACRR	BERT (fine-tuned)	**[BCVG] 0.4559**	**[BCVG] 0.5150**	**[BCVGN] 0.3373**	**[BCVGN] 0.2656**
KNRM	GloVe	0.3408	0.3871	[B] 0.2448	0.1755
KNRM	ELMo	[C] 0.3517	[CG] 0.4089	0.2227	0.1689
KNRM	BERT	[BCG] 0.3817	[CG] 0.4318	[B] 0.2525	[B] 0.1944
KNRM	BERT (fine-tuned)	[BCG] 0.4221	[BCVG] 0.4858	[BCVG] 0.3287	[BCVG] 0.2557
CEDR-KNRM	BERT (fine-tuned)	**[BCVGN] 0.4667**	**[BCVGN] 0.5381**	**[BCVG] 0.3469**	**[BCVG] 0.2772**
DRMM	GloVe	0.2892	0.3040	0.2215	0.1603
DRMM	ELMo	0.2867	0.3137	[B] 0.2271	0.1762
DRMM	BERT	0.2878	0.3194	[BG] 0.2459	[BG] 0.1977
DRMM	BERT (fine-tuned)	[CG] 0.3641	[CG] 0.4135	[BG] 0.2598	[B] 0.1856
CEDR-DRMM	BERT (fine-tuned)	**[BCVGN] 0.4587**	**[BCVGN] 0.5259**	**[BCVGN] 0.3497**	**[BCVGN] 0.2621**

rate when using static GloVe vectors is orders of magnitude faster than when using the contextualized representations, with BERT outperforming ELMo because it uses the more efficient Transformer instead of an RNN. In an attempt to improve the running time of these systems, we propose limiting the number of layers processed by the model. The reasoning behind this approach is that the lower the layer, the more abstract the matching becomes, perhaps becoming less useful for ranking. We show the runtime and ranking performance of PACRR when only processing only up to a given layer in Figure 2(b). It shows that most of the performance benefits can be achieved by only running BERT through layer 5; the performance is comparable to running the full BERT model, while running more than twice as fast. While we acknowledge that our research code is not completely optimized, we argue that this approach is generally applicable because the processing of these layers are sequential, query-dependent, and dominate the processing time of the entire model. This approach is a simple time-saving measure.

4 CONCLUSION

We demonstrated that contextualized word embeddings can be effectively incorporated into existing neural ranking architectures and suggested an approach for improving runtime performance by limiting the number of layers processed.

REFERENCES

[1] Zhuyun Dai, Chenyan Xiong, James P. Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *WSDM*.

[2] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR*.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805* (2018).

[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*.

[6] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. In *WSDM*.

[7] Samuel Huston and W Bruce Croft. 2014. Parameters learned in the comparison of retrieval models using term dependencies. *Technical Report* (2014).

[8] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[9] Kui-Lam Kwok, Laszlo Grunfeld, H. L. Sun, and Peter Deng. 2004. TREC 2004 Robust Track Experiments Using PIRCS. In *TREC*.

[10] Nut Limsopatham, Richard McCreadie, M-Dyaa Albakour, Craig MacDonald, Rodrygo L. T. Santos, and Iadh Ounis. 2012. University of Glasgow at TREC 2012: Experiments with Terrier. In *TREC*.

[11] Xitong Liu, Peilin Yang, and Hui Fang. 2014. Entity Came to Rescue - Leveraging Entities to Minimize Risks in Web Search. In *TREC*.

[12] Ryan McDonald, Yichun Ding, and Ion Androutsopoulos. 2018. Deep Relevance Ranking using Enhanced Document-Query Interactions. In *EMNLP*.

[13] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *SIGIR*.

[14] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *CoRR* abs/1901.04085 (2019).

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*.

[16] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

[17] Fiana Raiber and Oren Kurland. 2013. The Technion at TREC 2013 Web Track: Cluster-based Document Retrieval. In *TREC*.

[18] Corby Rosset, Damien Jose, Gargi Ghosh, Bhaskar Mitra, and Saurabh Tiwary. 2018. Optimizing Query Evaluations Using Reinforcement Learning for Web Search. In *SIGIR*.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.

[20] Chenyan Xiong, Zhuyun Dai, James P. Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *SIGIR*.

[21] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *SIGIR*.

[22] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-End Open-Domain Question Answering with BERTserini. *CoRR* abs/1901.04085 (2019).

[23] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks?. In *NIPS*.

[24] Hamed Zamani, Mostafa Dehghani, Fernando Diaz, Hang Li, and Nick Craswell. 2018. SIGIR 2018 Workshop on Learning from Limited or Noisy Data for Information Retrieval. In *SIGIR*.