CEDR: Contextualized Embeddings for Document Ranking

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

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

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

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 modes (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.1

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 input 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 (Contextualized 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 the term representations.

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.2

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 \( \{q_1, q_2, \ldots, q_{|Q|}\} \), and let \( D \) be a document consisting of terms \( \{d_1, d_2, \ldots, d_{|D|}\} \). Let \( \text{ranker}(Q, D) \in \mathbb{R} \) be a function that returns a real-valued relevance estimate for the document to the query. Neural relevance ranking architectures generally use a similarity matrix as input \( S \in \mathbb{R}^{(|Q| \times |D|)} \), where each cell represents a similarity score between the query and document: \( S_{i,j} = \text{sim}(q_i, d_j) \). 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

1Accepted to SIGIR 2019, camera ready to follow.
2https://github.com/Georgetown-IR-Lab/cedr
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 a single 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 $S_{Q,D} \in \mathbb{R}^{L \times |Q| \times |D|}$ where $L$ is the number of layers in the model. This is akin to the channel in image processing. Let $\text{context}_{Q,D}(t, l) \in \mathbb{R}^{D}$ return the contextualized language model representation for token $t$ in layer $l$, given the context of $Q$ and $D$. Given these definitions, let the contextualized representation be:

$$S_{Q,D}(l, q, d) = \cos(\text{context}_{Q,D}(q, l), \text{context}_{Q,D}(d, l))$$

(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 models (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. For models using dense combination (e.g., PACRR, KNRM), we propose concatenating the [CLS] vector to the model’s signals (e.g., n-gram or kernel scores). 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 5 folds from [7] with training and validation sets rotating in a round-robin fashion. 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 5³, WebTrack 2009–12 use ClueWeb09⁴, and WebTrack 2013–14 uses ClueWeb12⁵ as document collections. We evaluate the results using the official metrics for each dataset: nDCG@20 / P@20 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 same data using the Vanilla BERT classifier (see baseline below), and further fine-tuned when training the ranker itself.

**Training and optimization.** We trained all models using pairwise loss in line with prior work that has found it effective [2]. 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 use the validation metric P@20 on Robust and nDCG@20 on WebTrack to select the best-performing model on the validation set. At inference time, the top 100 documents from BM25 are re-ranked. The neural re-ranking approach is more practical than ranking all qrels, as demonstrated by the fact that major search engines employ a pipeline approach [18]. All models are trained using Adam [8] with a learning rate of 0.001

---

³520k documents; https://trec.nist.gov/data_disks.html

⁴1B web pages, https://lemurproject.org/clueweb09/php

⁵733M web pages, https://lemurproject.org/clueweb12/
with BERT layers are trained at a rate of 2e-5. Following prior work [6], all documents are truncated to the first 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.7 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 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 automatic topic-only TREC submission for each track in terms of nDCG@20. We use uogTrA44xu for WT12 ([10], a learning-to-rank based run), c1ustermfaE for WT13 ([17], a clustering-based approach), UDInfWebAX for WT14 ([11], an entity expansion approach), and pircRB84xt for Robust04 ([9], a query expansion approach using Google search results).8

- ConvKNRM [1], our implementation with the same training pipeline as the evaluation models.

- Each evaluation model when using GloVe [15] vectors.9

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 and ConvKNRM baselines on its own. This is encouraging, and shows the matching power of the Vanilla BERT model. When using contextualized language term representations without tuning, PACRR and DRMM performance is comparable to that of GloVe, 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, and nearly always outperforming the non-joint versions. 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.

---

7Pilot 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].

8k1 in 0.1–4 (by 0.1), b in 0.1–1 (by 0.1), SDM weights in 0–1 (by 0.05).

9We acknowledge limitations of the TREC experimentation environment. GloVe. 42B. 300d, https://nlp.stanford.edu/projects/glove/

---

10Running time measured on commodity hardware: single GeForce GTX 1080 Ti GPU, pairs per-processed and in memory.
We demonstrated that state-of-the-art re-ranking performance 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 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 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 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...
[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 BERTserni. 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.