Applying BERT to Document Retrieval with Birch

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and **Jimmy Lin**
David R. Cheriton School of Computer Science
University of Waterloo

Abstract

We present Birch, a system that applies BERT to document retrieval via integration with the open-source Anserini information retrieval toolkit to demonstrate end-to-end search over large document collections. Birch implements simple ranking models that achieve state-of-the-art effectiveness on standard TREC newswire and social media test collections. This demonstration focuses on technical challenges in the integration of NLP and IR capabilities, along with the design rationale behind our approach to tightly-coupled integration between Python (to support neural networks) and the Java Virtual Machine (to support document retrieval using the open-source Lucene search library). We demonstrate integration of Birch with an existing search interface as well as interactive notebooks that highlight its capabilities in an easy-to-understand manner.

1 Introduction

The information retrieval community, much like the natural language processing community, has witnessed the growing dominance of approaches based on neural networks. Applications of neural networks to document ranking usually involve multi-stage architectures, beginning with a traditional term-matching technique (e.g., BM25) over a standard inverted index, followed by a reranker that rescores the candidate list of documents (Asadi and Lin, 2013).

Researchers have developed a panoply of neural ranking models—see Mitra and Craswell (2019) for a recent overview—but there is emerging evidence that BERT (Devlin et al., 2019) outperforms previous approaches to document retrieval (Yang et al., 2019c; MacAvaney et al., 2019) as well as search-related tasks such as question answering (Nogueira and Cho, 2019; Yang et al., 2019b).

We share with the community Birch,[1] which integrates the Anserini information retrieval toolkit[2] with a BERT-based document ranking model that provides an end-to-end open-source search engine. Birch allows the community to replicate the state-of-the-art document ranking results presented in Yilmaz et al. (2019) and Yang et al. (2019c). Here we summarize those results, but our focus is on system architecture and the rationale behind a number of implementation design decisions, as opposed to the ranking model itself.

2 Integration Challenges

The problem we are trying to solve, and the focus of this work, is how to bridge the worlds of information retrieval and natural language processing from a software engineering perspective, for applications to document retrieval. Following the standard formulation, we assume a (potentially large) corpus D that users wish to search. For a keyword query Q, the system's task is to return a ranked list of documents that maximizes a retrieval metric such as average precision (AP). This stands in contrast to reading comprehension tasks such as SQuAD (Rajpurkar et al., 2016) and many formulations of question answering today such as WikiQA (Yang et al., 2015) and the MS MARCO QA Task (Bajaj et al., 2018), where there is no (or minimal) retrieval component. These are better characterized as "selection" tasks on (pre-determined) text passages.

Within the information retrieval community, there exists a disconnect between academic researchers and industry practitioners. Outside of a few large organizations that deploy custom infrastructure (mostly commercial search engine companies), Lucene (along with the closely-related

[1] http://birchir.io/
[2] http://anserini.io/

Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations), pages 19–24
Hong Kong, China, November 3 – 7, 2019.

projects Solr and Elasticsearch) has become the de facto platform for building real-world search applications, deployed at Twitter, Netflix, eBay, and numerous other organizations. However, many researchers still rely on academic systems such as Indri[3] and Terrier,[4] which are mostly unknown in real-world production environments. This gap hinders technology transfer and the potential impact of research results.

Even assuming Lucene as a "common denominator" that academic researchers learn to adopt, there is still one technical hurdle: Lucene is implemented in Java, and hence runs on the Java Virtual Machine (JVM). However, most deep learning toolkits today, including TensorFlow and PyTorch, are written in Python with a C++ backend. Bridging Python and the JVM presents a technical challenge for NLP/IR integration.

3 Birch

3.1 Anserini

Anserini (Yang et al., 2017, 2018) represents an attempt to better align academic researchers with industry practitioners by building a research-focused toolkit on top of the open-source Lucene search library. Further standardizing on a common platform within the academic community can foster greater replicability and reproducibility, a growing concern in the community (Lin et al., 2016).

Already, Anserini has proven to be effective and has gained some traction: For example, Nogueira and Cho (2019) used Anserini for generating candidate documents before applying BERT to ranking passages in the TREC Complex Answer Retrieval (CAR) task (Dietz et al., 2017), which led to a large increase in effectiveness. Yang et al. (2019b) also combined Anserini and BERT to demonstrate large improvements in open-domain question answering directly on Wikipedia.

3.2 Design Decisions

The architecture of Birch is shown in Figure 1, which codifies a two-stage pipeline architecture where Anserini is responsible for retrieval, the output of which is passed to a BERT-based reranker. Since our research group has standardized on PyTorch, the central challenge we tackle is: How do we integrate the deep learning toolkit with Anserini?

[3] https://www.lemurproject.org/

[4] http://terrier.org/

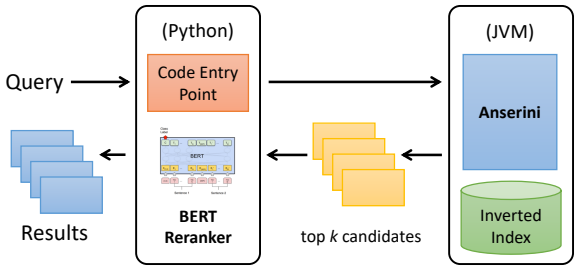

Figure 1: Architecture of Birch, illustrating a tight integration between Python and the Java Virtual Machine. The main code entry point is in Python, which calls Anserini for retrieval; candidate documents from Anserini are then reranked by our BERT models.

At the outset, we ruled out "loosely-coupled" integration approaches: For example, passing intermediate text files is not a sustainable solution in the long term. It is not only inefficient, but interchange formats frequently change (whether intentionally or accidentally), breaking code between multiple components. We also ruled out integration via REST APIs for similar reasons: efficiency (overhead of HTTP calls) and stability (imperfect solutions for enforcing API contracts, particularly in a research environment).

There are a few options for the "tightly-coupled" integration we desired. In principle, we could adopt the Java Virtual Machine (JVM) as the primary code entry point, with integration to the Torch backend via JNI, but this was ruled out because it would create two separate code paths (JVM to C++ for execution and Python to C++ for model development), which presents maintainability issues. After some exploration, we decided on Python as the primary development environment, integrating Anserini using the Pyjnius Python library[5] for accessing Java classes. The library was originally developed to facilitate Android development in Python, and allows Python code to directly manipulate Java classes and objects. Thus, Birch supports Python as the main development language (and code entry point, as shown in Figure 1), connecting to the backend JVM to access retrieval capabilities.

3.3 Models

Our document ranking approach is detailed in Yilmaz et al. (2019) and Yang et al. (2019c). We follow Nogueira and Cho (2019) in adapting BERT for binary (specifically, relevance) classification over text. Candidate documents from Anserini

[5] https://pyjnius.readthedocs.io/

Model	2011		2012		2013		2014	
	AP	**P@30**	**AP**	**P@30**	**AP**	**P@30**	**AP**	**P@30**
QL	0.3576	0.4000	0.2091	0.3311	0.2532	0.4450	0.3924	0.6182
RM3	0.3824	0.4211	0.2342	0.3452	0.2766	0.4733	0.4480	0.6339
MP-HCNN (Rao et al., 2019)	0.4043	0.4293	0.2460	0.3791	0.2896	0.5294	0.4420	0.6394
BiCNN (Shi et al., 2018)	0.4293	0.4728	0.2621	0.4147	0.2990	0.5367	0.4563	0.6806
Birch	**0.4697**	**0.5040**	**0.3073**	**0.4356**	**0.3357**	**0.5656**	**0.5176**	**0.7006**

Table 1: Results on test collections from the TREC Microblog Tracks, comparing BERT with selected neural ranking models. The first two blocks of the table contain results copied from Rao et al. (2019).

are processed individually. As model input, we concatenate the query Q and document D into a text sequence [[CLS], Q, [SEP], D, [SEP]], and then pad each text sequence in a mini-batch to N tokens, where N is the maximum length in the batch. The [CLS] vector is then taken as input to a single layer neural network. Starting from a pre-trained BERT model, we fine-tune with existing relevance judgments using cross-entropy loss. BERT inference scores are then combined with the original retrieval scores, in the simplest case, using linear interpolation.

In this simple approach, long documents pose a problem since BERT wasn't specifically designed to perform inference on long input texts. We present a simple solution: inference is applied over each sentence in a candidate document and sentence-level evidence is aggregated for ranking documents as follows:

$$S_f = a \cdot S_{doc} + (1 - a) \cdot \sum_{i=1}^{n} w_i \cdot S_i \qquad (1)$$

where S_{doc} is the original document score and S_i is the i-th top-scoring sentence according to BERT; a and w_i's are parameters that need to be learned. In practice, we only consider up to the three top-scoring sentences in each document.

The intuition behind this approach comes from Zhang et al. (2018b,a), who found that the "best" sentence or paragraph in a document provides a good proxy for document relevance. From a different perspective, we are essentially implementing a form of passage retrieval.

4 Retrieval Results

4.1 TREC 2011–2014 Microblog Tracks

As originally reported in Yang et al. (2019c), Birch was evaluated on tweet test collections from the TREC Microblog Tracks, 2011 to 2014 (Lin et al., 2014). Since tweets are short, relevance judgments can be directly used to fine-tune the BERT model (Section 3.3). For evaluation on each year's dataset, we used the remaining years for fine-tuning, e.g., tuning on 2011–2013 data, testing on 2014 data. Additional details on the fine-tuning strategy and experimental settings are described in Yang et al. (2019c).

At retrieval (inference) time, query likelihood (QL) with RM3 relevance feedback (Nasreen et al., 2004) was used to provide the initial pool of candidates (to depth 1000). Since tweets are short, we can apply inference over each candidate document in its entirety. The interpolation weight between the BERT scores and the retrieval scores was tuned on the validation data.

Experimental results are shown in Table 1, where we present average precision (AP) and precision at rank 30 (P@30), the two official metrics of the evaluation (Ounis et al., 2011). The first two blocks of the table are copied from Rao et al. (2019), who compared bag-of-words baselines (QL and RM3) to several popular neural ranking models as well as MP-HCNN, the model they introduced. The results of Rao et al. (2019) were further improved in Shi et al. (2018); in all cases, the neural models include interpolation with the original document scores. We see that Birch yields a large jump in effectiveness across all Microblog collections.

4.2 TREC 2004 Robust Track

In addition to searching short social media posts, we also examined a "traditional" document retrieval task over newswire articles. For this, we used the test collection from the TREC 2004 Robust Track (Voorhees, 2004), which comprises 250 topics over a newswire corpus of around 500K documents. Here, we provide a summary of Yilmaz et al. (2019), which contains more detailed

Model	AP	P@20	NDCG@20
BM25+RM3	0.2903	0.3821	0.4407
1S: BERT	0.3676	0.4610	0.5239
2S: BERT	**0.3697**	0.4657	0.5324
3S: BERT	0.3691	**0.4669**	**0.5325**

Table 2: Results on Robust04, where nS denotes combining scores from the top n sentences in a document.

descriptions of our approach and presents experiments on more test collections.

The additional challenge with ranking newswire articles is the lack of training data to fine-tune the BERT models, since relevance judgments are provided at the document level. That is, in the standard formulation of document ranking, a document is considered relevant if *any* part of it is relevant—but documents are typically longer than the lengths of text BERT was designed to handle. The surprising finding of Yilmaz et al. (2019) is that BERT models fine-tuned with the Microblog test collections in Section 4.1 can be *directly* applied to rank newswire documents, despite the differences in domain (social media posts vs. news articles). Furthermore, it appears that out-of-domain passage-level relevance judgments *fortuitously* available, such as the MS MARCO passage dataset (Bajaj et al., 2018) and the TREC CAR dataset (Dietz et al., 2017), are also beneficial.

Thus, it appears that BERT is able to learn *cross-domain*, *sentence-level* notions of relevance that can be exploited for ranking newswire documents. Table 2 presents an extract of results from Yilmaz et al. (2019) for Robust04, where we find that the best results are achieved by first fine-tuning on MS MARCO and then on the Microblog data. Scores from BERT are then combined with document scores (BM25+RM3) based on Eq (1). The notation "1S", "2S", and "3S" refer to aggregating scores from the top one, two, and three sentences, respectively. Including more sentences doesn't help and ranking is already quite good if we just consider the top-scoring sentence. This result, surprisingly, suggests that document ranking can be distilled into relevance prediction primarily at the sentence level. Based on the meta-analysis by Yang et al. (2019a), this is not only the highest known AP reported on the Robust04 dataset for neural models, but also exceeds the previous best known AP score of 0.3686, which is a non-neural method based on ensembles.

5 Demonstration

We demonstrate the integration of Birch with the search frontend from HiCAL (Abualsaud et al., 2018b) and interactive Google Colab notebooks.

5.1 HiCAL

The goal of the HiCAL system[6] is to help human assessors efficiently find as many relevant documents as possible in a large document collection to achieve high recall on a search task. The system comprises two main components: a Continuous Active Learning (CAL) model (Cormack and Grossman, 2014) and a search model. In the CAL model, a machine-learned classifier selects the most likely relevant document or paragraph for the assessor to judge; judgments are then fed back to retrain the classifier. In the current HiCAL implementation, Anserini provides the backend search capabilities.

For the TREC Common Core Tracks in 2017 and 2018, a small group of researchers used HiCAL to find and judge relevant documents. The runs generated based on their assessments achieved the highest AP scores among all the submitted runs for two consecutive years (Zhang et al., 2017; Abualsaud et al., 2018a). The effectiveness of the system was further demonstrated in Zhang et al. (2018a).

We further augment the Anserini backend for HiCAL with Birch in two ways: First, HiCAL can directly take advantage of improved rankings provided by BERT. Second, the top-scoring sentences can be highlighted in the document to aid in assessment. A sample screenshot is shown in Figure 2. For query 336 "black bear attacks" from Robust04, we show part of the highest-scoring document `LA081689-0039` with one of the top three sentences (according to BERT) highlighted.

5.2 Interactive Colab Notebooks

We present Google Colab[7] notebooks that make it possible for anyone to reproduce our end-to-end document retrieval pipeline in an interactive manner.[8] We make all our data and pre-trained models available, although users may also opt to rebuild them from scratch; the Colab GPU backend enables fine-tuning BERT models directly in the notebook environment.

[6] https://github.com/hical

[7] https://colab.research.google.com/

[8] To ensure long-term availability, sample notebooks are linked from the main Birch repository.

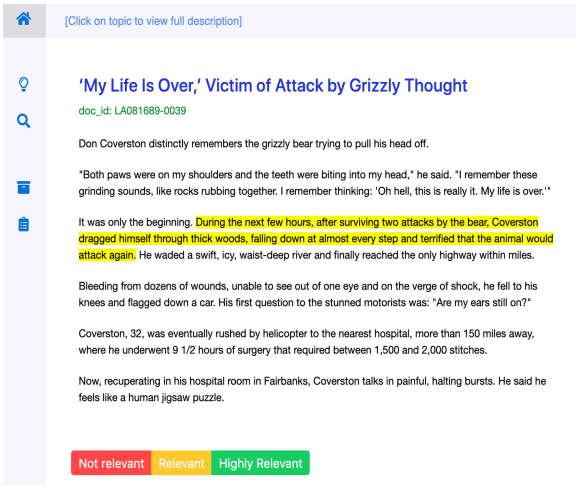

Figure 2: Screenshot of HiCAL using Birch to identify the most relevant sentences in a document retrieved for the query "black bear attacks".

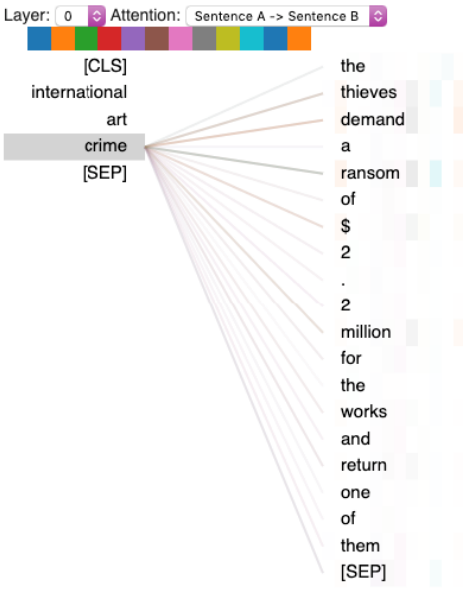

Figure 3: Screenshot of attention visualization for a sentence with a high BERT score from a document retrieved for the query "international art crime". Note the lack of exact matches with query terms.

Our notebooks are set up to allow relevance scores to be computed for an entire test collection in batch, and also to support interactive querying. When the user issues a query through the interactive notebook, candidate documents from the corpus are first retrieved using Anserini. A sentence-level dataset is created on the fly from the initial ranking by splitting each document into its constituent sentences. Each sentence is fed into our BERT model to obtain a relevance score. These relevance scores are then aggregated with document scores to rerank the candidate documents, per Eq (1). The notebook setting allows a user to step through each part of the process and examine intermediate results to gain a better understanding of our approach.

In addition, we have implemented two methods to visualize the relevant documents for a given query from a test collection, hopefully conveying even more insights. First, we generate a table that displays the document scores juxtaposed with the BERT scores of constituent sentences. Sentences with low document scores but high BERT scores (and vice versa) are highlighted, allowing the user to examine the relative contributions of exact term matching and semantic matching, as contributed by BERT. Second, we incorporate `bertviz`,[9] an open-source tool for visualizing attention in transformer models, to explore the interaction between multiple attention heads. In Figure 3, we show a sentence with a high BERT score in a document retrieved for query 322 "international art crime" from Robust04. Note that this sentence does *not* contain any of the query terms, but yet appears to be relevant. If we examine the attention visualization for the query term "crime", we see that the model attends to obviously-related terms like "thieves", "demand", and "ransom", illustrating the semantic knowledge that is captured in the BERT model.

[9] https://github.com/jessevig/bertviz

6 Conclusions

This paper describes the system architecture and motivation behind a straightforward application of BERT to document ranking, via sentence-level inference and simple score aggregation. With the implementation of this system, we have also overcome the technical challenge of integrating a Lucene-based backend on the JVM with PyTorch to enable development in an environment NLP researchers and practitioners are already familiar with. The fruits of our labor are released in an open-source system for the community to continue explorations in search-related tasks with BERT.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, and enabled by computational resources provided by Compute Ontario and Compute Canada.

References

M. Abualsaud, G. Cormack, N. Ghelani, A. Ghenai, M. Grossman, S. Rahbariasl, M. Smucker, and H. Zhang. 2018a. UWaterlooMDS at the TREC 2018 Common Core Track. In *TREC*.

M. Abualsaud, N. Ghelani, H. Zhang, M. Smucker, G. Cormack, and M. Grossman. 2018b. A system for efficient high-recall retrieval. In *SIGIR*, pages 1317–1320.

N. Asadi and J. Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *SIGIR*, pages 997–1000.

P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, M. Rosenberg, X. Song, A. Stoica, S. Tiwary, and T. Wang. 2018. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268v3*.

G. Cormack and M. Grossman. 2014. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *SIGIR*, pages 153–162.

J. Devlin, M. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

L. Dietz, M. Verma, F. Radlinski, and N. Craswell. 2017. TREC Complex Answer Retrieval overview. In *TREC*.

J. Lin, M. Crane, A. Trotman, J. Callan, I. Chattopadhyaya, J. Foley, G. Ingersoll, C. Macdonald, and S. Vigna. 2016. Toward reproducible baselines: The open-source IR reproducibility challenge. In *ECIR 2016*, pages 408–420.

J. Lin, M. Efron, Y. Wang, and G. Sherman. 2014. Overview of the TREC-2014 Microblog Track. In *TREC*.

S. MacAvaney, A. Yates, A. Cohan, and N. Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *SIGIR*, pages 1101–1104.

B. Mitra and N. Craswell. 2019. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, 13(1):1–126.

A. Nasreen, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M. Smucker, T. Strohman, H. Turtle, and C. Wade. 2004. UMass at TREC 2004: Novelty and HARD. In *TREC*.

R. Nogueira and K. Cho. 2019. Passage re-ranking with BERT. *arXiv:1901.04085*.

I. Ounis, C. Macdonald, J. Lin, and I. Soboroff. 2011. Overview of the TREC-2011 Microblog Track. In *TREC 2011*.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP 2016*, pages 2383–2392.

J. Rao, W. Yang, Y. Zhang, F. Ture, and J. Lin. 2019. Multi-perspective relevance matching with hierarchical ConvNets for social media search. *AAAI*, pages 232–240.

P. Shi, J. Rao, and J. Lin. 2018. Simple attention-based representation learning for ranking short social media posts. In *NAACL*, pages 2212–2217.

E. Voorhees. 2004. Overview of the TREC 2004 Robust Track. In *TREC 2004*, pages 52–69.

P. Yang, H. Fang, and J. Lin. 2017. Anserini: Enabling the use of Lucene for information retrieval research. In *SIGIR 2017*, pages 1253–1256.

P. Yang, H. Fang, and J. Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16.

W. Yang, K. Lu, P. Yang, and J. Lin. 2019a. Critically examining the "neural hype": weak baselines and the additivity of effectiveness gains from neural ranking models. In *SIGIR*, pages 1129–1132.

W. Yang, Y. Xie, A. Lin, X. Li, L. Tan, K. Xiong, M. Li, and J. Lin. 2019b. End-to-end open-domain question answering with BERTserini. In *NAACL Demo*, pages 72–77.

W. Yang, H. Zhang, and J. Lin. 2019c. Simple applications of BERT for ad hoc document retrieval. *arXiv:1903.10972*.

Y. Yang, W. Yih, and C. Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*, pages 2013–2018.

Z. Akkalyoncu Yilmaz, W. Yang, H. Zhang, and J. Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *EMNLP*.

H. Zhang, M. Abualsaud, N. Ghelani, A. Ghosh, M. Smucker, G. Cormack, and M. Grossman. 2017. UWaterlooMDS at the TREC 2017 Common Core Track. In *TREC*.

H. Zhang, M. Abualsaud, N. Ghelani, M. Smucker, G. Cormack, and M. Grossman. 2018a. Effective user interaction for high-recall retrieval: less is more. In *CIKM*, pages 187–196.

H. Zhang, G. Cormack, M. Grossman, and M. Smucker. 2018b. Evaluating sentence-level relevance feedback for high-recall information retrieval. *arXiv:1803.08988*.