University of Waterloo E-Thesis Template for LaTeX

by

Zeynep Akkalyoncu Yilmaz

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2019

Abstract

Standard bag-of-words term-matching techniques in document retrieval fail to exploit rich semantic information embedded in the document texts. One promising recent trend in facilitating context-aware semantic matching has been the development of massively pre-trained language models, culminating in BERT as its most popular example today. In this work, we propose adapting BERT as a neural reranker for document retrieval with large improvements on news articles. Two fundamental issues arise in applying BERT to "ad hoc" document retrieval on newswire collections: relevance judgements in existing test collections are provided only at the document level, and documents often exceed the length that BERT was designed to handle. To overcome these challenges, we compute and aggregate sentence-level relevance scores to rank documents. We solve the problem of lack of appropriate relevance judgements by leveraging sentence-level and passage-level relevance judgements available in collections from other domains to capture cross-domain notions of relevance. We demonstrate that models of relevance can be transferred across domains. By leveraging semantic cues learned across various domains, we propose a model that achieves state-of-the-art results across three standard TREC newswire collections. We explore the effects of cross-domain relevance transfer, and trade-offs between using document and sentence scores for document ranking. We also present an end-to-end document retrieval system that incorporates the open-source Anserini information retrieval toolkit, discussing the related technical challenges and design decisions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Document retrieval refers to the task of generating a ranking of documents from a large corpus D in response to a query Q. In a typical document retrieval pipeline, an inverted index is constructed in advance from the collection, which often comprises unstructured text documents, for fast access during retrieval. When the user issues a query, the query representation is matched against the index, computing a similarity score for each document. The top most relevant documents based on their closeness to the query are returned to the user in order of relevance. This procedure may be followed by a subsequent re-ranking stage where the candidate documents outputted by the previous step are further re-ranked in a way that maximizes some retrieval metric such as average precision (AP).

Document retrieval systems traditionally rely on term-matching techniques, such as BM25, to judge the relevance of documents in a corpus. More specifically, the more common terms a document shares with the query, the more relevant it is considered. As a result, these systems may fail to detect documents that do not contain exact query terms, but are nonetheless relevant. For example, consider a document that expresses relevant information in a way that cannot be resolved without external semantic analysis. Figure 1 displays

Query: international art crime
Text: The thieves demand a ransom of $2.2 million for the works and return one of them.

Figure 1.1: An example of a query-text pair from the TREC Robust04 collection where a relevant piece of text does not contain direct query matches.

one such query-text pair where words semantically close to the query need to be identified to establish relevance. This "vocabulary mismatch" problem represents a long-standing challenge in information retrieval. To put its significance into context, Zhao et al. [59] show in their paper on term necessity prediction that, statistically, the average query terms do not appear in as many as 30% of relevant documents in TREC 3 to 8 "ad hoc" retrieval datasets.

Clearly, the classic exact matching approach to document retrieval neglects to exploit rich semantic information embedded in the document texts. To overcome this shortcoming, a number of models such as Latent Semantic Analysis [12], which map both queries and documents into high-dimensional vectors, and measure closeness between the two based on vector similarity, has been proposed. This innovation has enabled semantic matching to improve document retrieval by extracting useful semantic signals. With the advent of neural networks, it has become possible to learn better distributed representations of words that capture more fine-grained semantic and syntatic information [30], [36]. More recently, massively unsupervised language models that learn context-specific semantic information from copious amounts of data have changed the tide in NLP research (e.g: ELMo [37], GPT-2 [40]). These models can be applied to various downstream tasks with minimal task-specific fine-tuning, highlighting the power of transfer learning from large pre-trained models. Arguably the most popular example of these deep language representation models is the Bidirectional Encoder Representations from Transformers (BERT) [13]. BERT has achieved state-of-the-art results across a broad range of NLP tasks from question answering to machine translation.

While BERT has enjoyed widespread adoption across the NLP community, its application in information retrieval research has been limited in comparison. Guo et al. [17] suggest that the lackluster success of deep neural networks in information retrieval may be owing to the fact that they often do not properly address crucial characteristics of the "ad hoc" document retrieval task. Specifically, the relevance matching problem in information retrieval and semantic matching problem in natural language processing are fundamentally different in that the former depends heavily on exact matching signals, query term importance and diverse matching requirements. In other words, it is crucial to strike a good balance between exact and semantic matching in document retrieval. For this reason, we employ both document scores based on term-matching and semantic relevance scores to determine the relevance of documents.

In this thesis, we extend the work of Yang et al. [55] by presenting a novel way to apply BERT to "ad hoc" document retrieval on long documents – particularly, newswire articles – with significant improvements. Following Nogueira et al. [33], we adapt BERT for binary relevance classification over text to capture notions of relevance. We then deploy the BERT-based re-ranker as part of a multi-stage architecture where an initial list of candidate documents is retrieved with a standard bag-of-words term matching technique. The BERT model is used to compute a relevance score for each constituent sentence, and the candidate documents are re-ranked by combining sentence scores with the original document score.

We emphasize that applying BERT to document retrieval on newswire documents is not trivial due to two main challenges: Firist of all, BERT has a maximum input length of 512 tokens, which is insufficient to accommodate the overall length of most news articles. To put this into perspective, a typical TREC Robust04 document has a median length of 679 tokens, and in fact, 66% of all documents are longer than 512 tokens. Secondly, most collections provide relevance judgements only at the document level. Therefore, we only know what documents are relevant for a given query, but not the specific spans within the document. To further aggravate this issue, a document is considered relevant as long as some part of it is relevant, and most of the document often has nothing to do with the query.

We address the abovementioned challenges by proposing two effective innovations: First, instead of relying solely on document-level relevance judgements, we aggregate sentence-level evidence to rank documents. As mentioned before, since standard newswire collections lack sentence level judgements to facilitate this approach, we instead explore leveraging sentence-level or passage-level judgements already available in collections in other domains, such as tweets and reading comprehension. To this end, we fine-tune BERT models on these out-of-domain collections to learn models of relevance. Surprisingly, we demonstrate that models of relevance can indeed be successfully transferred across domains. It is important to note that the representational power of neural networks come at the cost of challenges in interpretability. For this reason, we dedicate a portion of this thesis to error analysis experiments in an attempt to qualify and better understand the cross-domain transfer effects. We also elaborate on our engineering efforts to ensure reproducibility and replicability, and the technical challenges involved in bridging the worlds of natural language processing and information retrieval from a software engineering perspective.

1.1 Contributions

The main contributions of this thesis can be summarized as follows:

- We present two innovations to successfully apply BERT to *ad hoc* document retrieval with large improvements: integrating sentence-level evidence to address the fact that BERT cannot process long spans posed by newswire documents, and exploiting cross-domain models of relevance for collections without sentence- or passage-level annotations. With the proposed model, we establish state-of-the-art effectiveness on three standard TREC newswire collections at the time of writing. Our results on Robust04 exceed the previous highest known score of 0.3686 [10] with a non-neural method based on ensembles, which has stood unchallenged for ten years.
- We explore through various error analysis experiments the effects of cross-domain relevance transfer with BERT as well as the contributions of BM25 and sentence scores to the final document ranking. Elaborate
- We release an end-to-end pipeline, Birch[1], that applies BERT to document retrieval over large document collections via integration with the open-source Anserini information retrieval toolkit. An accompanying Docker image is also included to ensure that anyone can easily deploy and test our system. We elaborate on the technical challenges in the integration of NLP and IR capabilities, and the rationale behind design decisions.

1.2 Thesis Organization

Add link to actual chapters The remainder of this thesis is organized in the following order: Chapter 2 reviews related work in neural document retrieval and transfer learning, particularly applications of BERT to document retrieval. Chapter 3 motivates the approach with some background information on the task, and introduces the datasets used for both training and evaluation as well as metrics. Chapter 4 proposes an end-to-end pipeline for document retrieval with BERT by elaborating on the design decisions and challenges. Chapter 5 describes the experimental setup, and presents the results on three newswire collections – Robust04, Core17 and Core18. Chapter 6 concludes the thesis by summarizing the contributions and discussing future work.

[1]https://github.com/castorini/birch

Chapter 2

Background and Related Work

2.1 Pretrained Language Models

Natural language processing tasks have traditionally been addressed with supervised learning on task-specific datasets. Due to the relatively small size of these datasets, training deep neural networks in this manner introduces the risk of overfitting on the training data, and lack of generalization across different datasets. With the increasing availability of large corpora, pretrained deep language models have been rapidly gaining traction among NLP researchers. Language model pretraining has proven extremely effective on many natural language processing tasks ranging from machine translation to reading comprehension. The underlying assumption in applying pretrained language models to downstream NLP tasks is that language modeling inherently captures many facets of language such as resolving long-term dependencies [25] and hierarchical patterns [16]. In general, pretrained language models can be applied to downstream tasks in one of two ways: feature-based and fine-tuning.

2.1.1 Feature-based Approaches

The feature-based approach, such as ELMo [37], employs deep pretrained representations learned with language modeling as additional features in task-specific architectures. This approach has the advantage of being easily incorporated into existing models with significant improvements in performance. ELMo [37] extends traditional word embeddings to learn context-sensitive features with a deep language model. Therefore, instead of taking the final layer of a deep LSTM (Long Short-Term Memory) as a word embedding, ELMo embeddings are learned as a function of *all* the internal states of a bidirectional deep LSTM language model. This method is motivated by a thread of work in NLP that suggests that the higher levels of a deep LSTM capture context [29] and meaning while the lower levels learn syntactic features well [6]. While traditional pretrained word embeddings like GloVe [36] cannot differentiate between homonyms, ELMo can as it generates different embeddings for them depending on their context. ELMo embeddings are constructed as a shallow concatenation of independently trained left-to-right and right-to-left LSTMs. Peters at al. [37] show that integrating deep contextualized embeddings learned with ELMo into task-specific architectures significantly improves over the original performance in six NLP tasks, including question answering on SQuAD [41] and sentiment analysis on the Stanford Sentiment Treebank (SST-5) [45].

2.1.2 Fine-tuning Approaches

The fine-tuning approach is inspired by the growing trend in transfer learning. These models are first pretrained with respect to a language modeling objective, and then applied to downstream NLP tasks by "freezing" their last layer, and "fine-tuning" on external data for the specific task with minimal task-specific parameters. This approach has been shown to greatly boosts the performance of many NLP tasks.

Radford et al. [40] claim that this phenomenon occurs because language models inherently capture many NLP tasks without explicit supervision. Therefore, they propose Generative Pretrained Transformers (GPT-2) to perform zero-shot task transfer on multiple sentence-level tasks from the GLUE benchmark [49] with impressive results. At the core of GPT-2 lies a multi-layer left-to-right transformer [47] decoder, with each layer consisting of a multi-head self-attention mechanism and fully connected feed-forward network [39]. The large capacity of the transformer is exploited by pretraining it on Google BookCorpus dataset [60] (800M words) where long contiguous spans of text allow the transformer to condition on long-range information.

To address the limitation of the unidirectional nature of GPT-2 [40], Bidirectional Encoder Representations from Transformers (BERT) [13] has introduced a novel way to pretrain bidirectional language models, and has since enjoyed widespread popularity across the NLP community. Standard language models cannot be conditioned on bidirectional context as this would cause the model to apply self-attention on the current token in a multi-layered context. However, BERT enables bidirectional language modeling by conditioning on both left and right context in all layers by employing a new pretraining objective called "masked language model" (MLM). Conceptually, MLM randomly masks some of the input tokens, i.e: 15% of tokens in each sequence, at random with the goal of predicting the masked tokens based only on their left and right context. The final hidden vectors corresponding to the masked tokens are then fed into a softmax layer over the vocabulary as in a standard language model. This objective allows the representation to fuse both left and right context, which is indispensable for token-level tasks such as question answering, according to the authors. Ablation studies confirm that the bidirectional nature of BERT is the single most important factor in BERT's performance. In additional to the novel language modeling approach, Devlin et al. [13] also propose a "next sentence prediction" task for applications that require an understanding of the relationship between two sentences, such as question answering or language language inference. Essentially, this trains a binary classifier to determine whether or not one sentence follows another sentence.

The underlying model architecture of BERT is a multi-layer bidirectional transformer [47]: The larger BERT model has 24 layers each with 1024 hidden nodes, and 16 self-attention heads in total. It is pretrained on the union of Google BookCorpus [60] (800M words) and English Wikipedia (2,500M words). The input representation for BERT is formed by concatenating the token with segment and position embeddings. Furthermore, the input may contain a single sentence or a sentence pair separated by the meta-token `[SEP]`, i.e: separator. Each sequence is prepended with `[CLS]`, corresponding to the "class" meta-token, whose final hidden state can be used for classification tasks. The words are represented

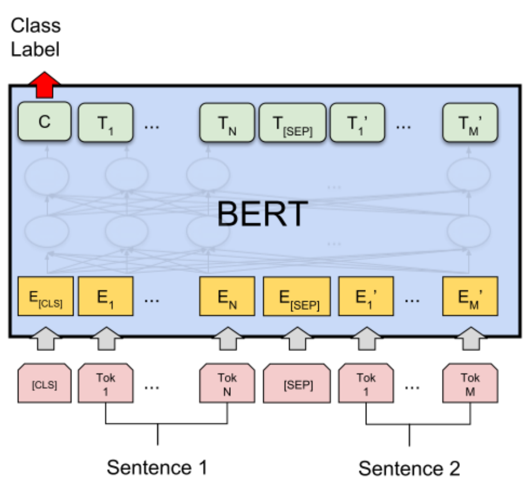

Figure 2.1: The architecture combining BERT an additional output layer for the sentence pair classification task, where E represents the input embedding for each sentence and T_i the contextual representation of token i. Adapted from Delvin et al. [13].

with WordPiece embeddings [50] with a vocabulary of 30,000 tokens. Originally proposed for segmentation problem in Japanese and Korean, the WordPiece model is used to divide words into small sub-word units in order to handle rare or out-of-vocabulary words more effectively. Positional embeddings are learned – not hard-coded – for up to 512, which is the maximum input size allowed by BERT.

To fine-tune BERT for classification tasks, a single-layer neural network is added on top of BERT with the class label as the input, and label probabilities are computed with softmax. The parameters of the additional layer and BERT are fine-tuned jointly to maximize the log-probability of the correct label. For span-level and token-level prediction tasks, the final step needs to be modified to account for multiple tokens. Figure 2.1 visualizes the model for fine-tuning BERT for the "sentence pair classification" task that takes two sentences separated by a `[SEP]` token as the input.

BERT has been applied to a broad range of NLP tasks from sentence classification to sequence labeling with impressive results. Most relevant to the task of document retrieval, applications of BERT include BERTserini by Yang et al. [54] which integrated BERT with Anserini for question answering over Wikipedia by fine-tuning BERT on SQuAD, and Nogueira et al. [33] who adopted BERT for passage reranking over MS MARCO.

2.2 Document Retrieval

2.2.1 Non-neural Document Retrieval

Traditional document retrieval techniques have evolved from the simple Boolean model to probabilistic models such as the Binary Independence Model over time to increase matching effectiveness. While these methods perform reasonably well for consistently short text like titles or abstracts, they have fallen short with the development of modern text collections with highly variable lengths. To this end, Okapi BM25 (commonly dubbed BM25) was developed as a bag-of-words ranking function that is sensitive to both term frequency and document length without introducing too many additional parameters [22]. Intuitively, BM25 pays more attention to the rarer terms in a query by increasing their term weight while dampening the matching signal for words that occur too frequently in a document with a term saturation mechanism. Term weights are also normalized with respect to the document length.

In addition to a term weighting scheme, query expansion has also been found to improve retrieval effectiveness by increasing recall. Unlike manual relevance feedback, pseudo

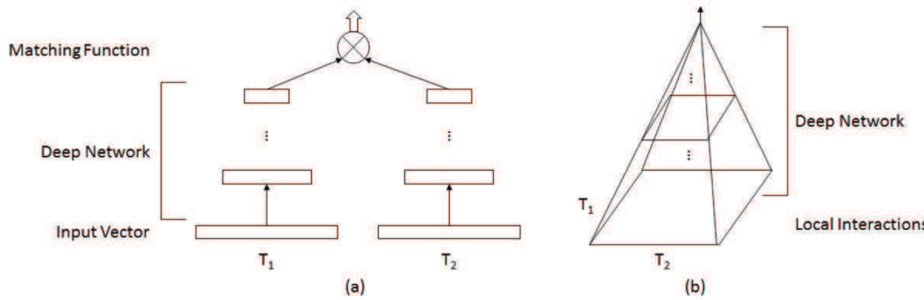

Figure 2.2: The two types of deep matching architectures: representation-focused (a) and interaction-focused (b). Adapted from Guo et al. [18].

relevance feedback allows for automatic local analysis without extended interaction with the user. RM3 is one such pseudo-relevance feedback mechanism where the original query is expanded by adding the top terms that appear in the contents of top k most relevant BM25 documents. While this method still relies on exact matching of query terms, it partly relieves the problem of synonymy. For instance, a query that contains the term "assistance" may be augmented with another high-frequency term "support" in relevant documents, therefore extending the range of matching. One obvious danger, however, is that retrieval may be incorrectly biased towards certain terms that occur frequently in the most relevant documents, but are not directly relevant to the query. Despite their simplicity, well-tuned BM25+RM3 baselines achieve competitive effectiveness on TREC collections [24].

2.2.2 Neural Document Retrieval

With the impressive results achieved by neural networks in many areas such as computer vision and natural language processing, document retrieval, too, has witnessed a shift from non-neural methods to neural methods over the last few years. Neural models have especially been instrumental in facilitating semantic matching in document retrieval. Neural models developed to address the deep matching problem in document retrieval can be divided into two broad categories based on their underlying architecture: representation-based and interaction-based. The high-level differences between these architectures can be observed in Figure 2.2.

Representation-based Models

Representation-based approaches first construct a representation from the input vectors for the query and document with a deep neural network, and then perform matching between these representations (see Figure 2.2). One set of such models loans the concept of word embeddings from natural language processing to represent query and documents. This paradigm represents words as low-dimensional continuous feature vectors that embody hidden semantic or syntactic dependencies. Some of the most popular pretrained English word embeddings include word2vec [30] trained on Google News, GloVe [36] on Common Crawl, Wikipedia and Twitter, and fastText [9] on English webcrawl and Wikipedia. These word embeddings can be learned from scratch for a specific corpus or pretrained over large corpora and reused with significant improvements over the former option [46]. There has also been some effort in learning word embeddings to directly capture relevance matching [56, 15] rather than linguistic features as in word2vec [30] or GlovE [36]. Word embeddings are commonly used as input to many representation-based retrieval models.

Other representation-based architectures explore alternative ways to represent text for effective retrieval. DSSM (short for Deep Structured Semantic Models) [20] extends previously dominant latent semantic models to deep semantic matching for web search by projecting query and documents into a common low-dimensional space. In order to accommodate a large vocabulary required by the task, the text sequences are mapped into character-level trigrams with a word hashing layer before computing a similarity matrix through dot product and softmax layers. While shown effective on a private dataset comprised of log files of a commercial search engine, DSSM requires too much training data to be effective. Moreover, DSSM cannot match synonyms because it is based on the specific composition of words and not semantic proximity. C-DSSM [44] was proposed as an extension to DSSM by replacing the multi-layer perceptron with a convolutional layer to devise semantic vectors for search queries and Web document. By performing a max pooling operation to extract local contextual information at the n-gram level, a global vector representation is formed from the local features. Shen et al. [44] demonstrate that both local and global contextual features are necessary for semantic matching for Web search. While C-DSSM improves over DSSM by exploiting the context of each trigram, it still suffers from most of the same issues listed above.

Interaction-based Models

Interaction-based approaches capture local matching signals, and directly compute word-word similarity between query and document representations. In contrast to more shallow representation-based approaches, this setup allows the deep neural network to learn more complex hierarchical matching patterns across multiple layers. Some notable examples of these architectures include DRMM [18], KNRM [51] and DUET [32].

DRMM [18], which stands for Deep Relevance Matching Model, maps variable-length local interactions of query and document into a fixed-length matching histogram. A feed forward matching network is used to learn hierarchical matching patterns from the histogram representation, and a matching score is computed for each term. An overall matching score is obtained by aggregating the scores from each query term with a term gating network. KNRM [51] similarly calculates the word-word similarities between query and document embeddings, but converts word-level interactions into ranking features with a novel kernel pooling technique. Specifically, a feature vector for each word in the query is constructed from the similarity matrix with k-max pooling. Ranking features are combined to form a final ranking score through a learning-to-rank layer. Unlike DRMM and KNRM, the goal of DUET [32] is to employ both local and distributed representations, therefore leveraging both exact matching and semantic matching signals. DUET is composed of two separate deep neural networks, one to match the query and the document using a one-hot representation, and another using learned distributed representations, which are trained jointly. The former estimates document relevance based on exact matches of the query terms in the document by computing an interaction matrix from one-hot encodings. The latter instead performs semantic matching by computing the element-wise product between the query and document embeddings. Their approach significantly outperform traditional baselines for web search with lots of clickthrough logs.

Contextualized Language Models

While the models introduced in Section 2.2.2 successfully leverage semantic information to varying degrees, they are limited by the size and variability of available training data. Ideally, these models would be trained on a large number of semantically and syntactically varied labeled query-document pairs; however, it is impractical to automatically gather a sufficient number of such training samples at scale. Instead massively pretrained unsupervised language models hold promises for obtaining better query and document representations, and therefore, achieving unprecedented effectiveness at semantic matching without the need for more relevance information. Section 2.1 outlines some of the most popular unsupervised language models that form the basis of effective retrieval architectures. In general, these language models are deployed as re-rankers over an initial list of candidate documents retrieved with traditional term-matching techniques in Section 2.2.1.

Modeling relevance requires an understanding of the relationship between two text sequences, e.g: the query and the document. Clearly, traditionally language modeling does not suffice to capture such a relationship. Fortunately, BERT facilitates such relevance classification by pre-training a binary next sentence prediction task based on its masking language model approach as discussed in Section 2.1. However, it is still not trivial to apply BERT to document retrieval because BERT was not designed to handle long spans of text, such as documents, given a maximum input length of 512 tokens. Partly because of this inherent challenge, the majority of work on re-ranking with BERT has focused on passage re-ranking instead of document re-ranking.

Notably, Nogueira et al. [33] proposed to re-rank MS MARCO passages based on a simple re-implementation of BERT to learn a model of relevance, outperforming the previous state of the art by 27% in MRR@10 and replacing the previous top entry in the leaderboard of the MS MARCO passage retrieval task. Our neural model is inspired by the BERT re-implementation described in their paper. Padigela et al. [34] prioritize studying the reasons behind the gains that come with re-ranking MS MARCO passages with BERT. To put re-ranking with BERT into perspective, they compare their BERT-based reranker to feature based learning to rank models such as RankSVM [21] and a number of neural kernel matching models such as KNRM [51] and Conv-KNRM [11], and conclude that fine-tuning BERT is substantially more effective than either neural model. They also test four hypotheses regarding the behavior of matching with BERT compared to BM25; specifically, with respect to term frequency and document length.

To our knowledge, Yang et al. [55] are the first to successfully apply BERT to "ad hoc" document retrieval. They demonstrate that BERT can be fine-tuned to capture relevance matching by following the "next sentence classification" task of BERT on the TREC Microblog Tracks where document length does not pose an issue. They further propose overcoming the challenge of long documents by applying inference on each individual sentence and combining the top scores to compute a final document score. Their approach is motivated by user studies by Zhang et al. [58] which suggest that the most relevant sentence or paragraph in a document provides a good proxy for overall document relevance. The work of Yang et al. [55] pave the way for future work that culminated in this thesis.

More recently, MacAvaney et al. [27] shifted focus from incorporating BERT as a reranker to using its representation capabilities to improve existing neural architectures. By computing a relevance matrix between the query and each candidate document at each layer of a contextualized language model – in particular, ELMo or BERT – they report a high score of NDCG@20 0.5381 on Robust04 by combining CEDR (Contextualized Embeddings for Document Ranking) [27] with KNRM [51]. They also propose a joint model that combines the classification mechanism of BERT into existing neural architectures to help benefit from both deep semantic matching with BERT *and* relevance matching with traditional ranking architectures.

A recent arXiv preprint by Qiao et al. [38] also examines the performance and behavior of BERT when used as a reranker for passage ranking on MS MARCO and for document ranking on the TREC Web Track. Their findings are consistent with those of Nogueira et al. [33] in that BERT outperforms previous neural models on the passage reranking task on MS MARCO. For ad hoc document ranking, they explore using BERT both as representation-based and interaction-based rankers and in combination with KNRM [51] and Conv-KNRM [11]. However, they find that their reranking TREC Web Track documents with BERT performs worse than Conv-KNRM and feature-based learning-to-rank models trained on user clicks in Bing's search log.

2.2.3 Comparison of Non-neural and Neural Methods

Despite growing interest in neural models for document ranking, researchers have recently voiced concern as to whether or not they have truly contributed to progress [24], at least in the absence of large amounts of behavioral log data only available to search engine companies. Some of the models discussed in this section are designed for the web search task where a variety of other signals are available, such as large amounts of log data and the webgraph. However, this is not the case for "ad hoc" document retrieval where the only available data is the document text, which is the main focus of this thesis. The SIGIR Forum piece by Lin [24] also echoes the general skepticism concerning the empirical rigor and contributions of machine learning applications in Lipton et al. [26] and Sculley at al. [43]. In particular, Lin et al. [24] lament that comparisons to weak baselines sometimes inflate the merits of certain neural information retrieval methods.

To rigorously study the current state of document retrieval literature, Yang et al. [53] recently conducted a thorough meta-analysis of over 100 papers that report results on the TREC Robust 2004 Track. Their findings are illustrated in Figure 2.3 where the empty circles correspond to the baselines and filled circles to the best AP scores of each paper. The solid black line represents the best submitted run at AP 0.333, and the dotted black line the median TREC run at AP 0.258. The other line is a RM3 baseline run with default parameters from the Anserini open-source information retrieval toolkit [52] at AP 0.3903.

The untuned RM3 baseline is more effective than 60% of all studied papers, and 20% of them report results below the TREC median. More surprisingly, only six of the papers report AP scores higher than the TREC best, with the highest being by Cormack et al. [10] in 2009 at AP 0.3686. Their approach is based on building an ensemble of multiple TREC runs with reciprocal rank fusion. Among the neural models, the highest encountered score is by Zamani et al. [57] in 2018 at AP 0.2971. Deviating from the dominant approach of deploying neural models as rerankers, Zamani et al. [57] propose a standalone neural ranking model to learn a latent representation for each query and document, which is sparse enough to construct an inverted index for the entire collection. However, their reported result is still much lower than the TREC best and far below the best reported result of Cormack et al. [10]. Moreover, about half of the neural papers compare their results to a baseline *below* the TREC median, which is consistent with the claims of Lin et al. [24]. Overall, Figure 2.3 exhibits no clear upward trend in terms of AP on Robust04 from 2005 to 2018.

TREC Common Core 2017 (Core17) [1] and 2018 (Core18) [2] are two of the more recent document collections that we evaluate our models on, which are not nearly as well-studied as Robust04 yet. Excluding runs that make use of past labels or require human intervention, the TREC best run on Core17 is `umass baselnrm` at AP 0.275 and on Core18 `uwmrg` at AP 0.276. To our knowledge, Neural Vector Spaces for Unsupervised Information Retrieval by Van Gysel et al. [19] represents the only major neural model evaluated on Core17. While their approach has the advantage of not requiring supervised relevance judgements, their reported results are quite low. Otherwise, evaluation of neural retrieval

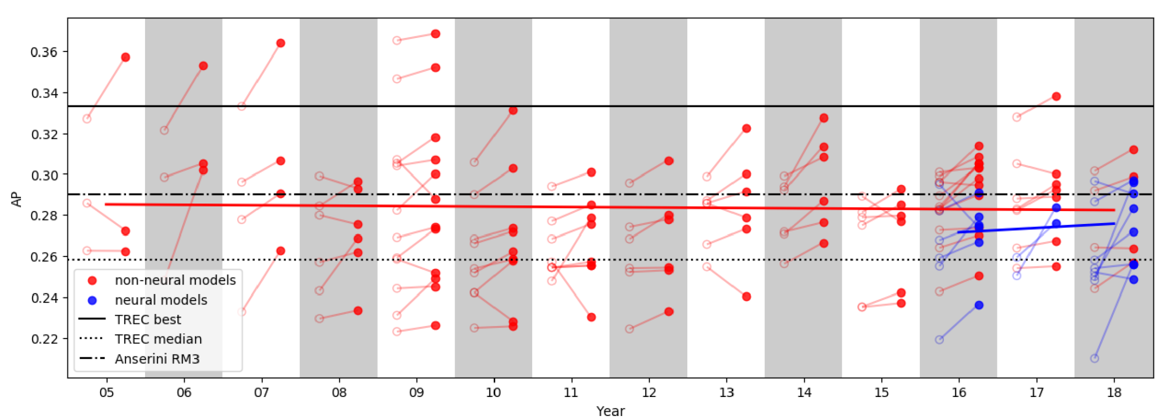

Figure 2.3: Visualization of best AP scores on Robust04 for 108 papers based on non-neural and neural approaches. Adapted from Yang et al. [53].

methods on both Core17 and Core18 has been limited.

2.3 Evaluation Metrics

Evaluation in information retrieval relies on the distinction between "relevant" and "irrelevant" documents with respect to an information need as expressed by a query. A number of automatic evaluation metrics has been formalized specifically for ranking tasks, some of which are described below.

2.3.1 Mean Average Precision (MAP)

Precision specifies what fraction of a set of retrieved documents is in fact relevant for a given query q. By extension, average precision (AP) expresses the average of the precision values obtained for the set of top k documents for the query. Support that $D = \{d_1, ..., d_{m_j}\}$ is the set of all relevant documents for a query q_j, then AP can be formulated as:

$$AP = \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{2.1}$$

where R_{jk} represents the set of top k ranked retrieval results.

The respective AP for each query $q_j \in Q$ can be aggregated to obtain mean average precision (MAP) for the overall retrieval effectiveness in the form of a single-figure measure of quality across various recall levels:

$$MAP = \frac{\sum_{j=1}^{|Q|} AP}{Q} = \frac{1}{Q} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{2.2}$$

MAP is known to have especially good discrimination and stability compared to other evaluation metrics, which makes it the ideal choice for large text collections [28]. It is hence one of the standard metrics among the TREC community.

2.3.2 Precision at k (P@k)

While MAP factors in precision at all recall levels, certain applications may have a distinctly different notion for ranking quality. Particularly in the case of web search, the user often only cares about the results on the first page or two. This restriction essentially requires measuring precision at fixed low levels of retrieved results, i.e: top k documents – hence the name for the metric "precision at k". On the one hand, it eliminates the need for any estimate of the size of the set of relevant documents because it is only concerned with the top documents. However, it also produces the least stable results out of all evaluation metrics. Moreover, precision at k does not average well because it is too sensitive to the total number of relevant documents for a query.

2.3.3 Normalized Discounted Cumulative Gain (NDCG@k)

Cumulative gain (CG) simply computes the sum of relevance labels for all the retrieved documents, treating the search results as an unordered set. However, since a highly relevant document is inherently more useful when it appears higher up in the search results, CG has been extended to discounted cumulative gain (DCG). DCG estimates the relevance of a document based on its rank among the retrieved documents. The relevance measure is accumulated from top to bottom, discounting the value of documents at lower ranks. NDCG at k measures DCG for the top k documents, normalizing by the highest possible value for a query; therefore, a perfect ranking yields NDCG equals 1.

NDCG is uniquely useful in applications with a non-binary notion of relevance, e.g: a spectrum of relevance. This makes NDCG comparable across different queries: The NDCG values for all queries can be averaged to reliably evaluate the effectiveness of a ranking algorithm for various information needs across a collection. Given a set of queries $q_j \in Q$ and relevance judgements R_{dj} for a document d:

$$NDCG(Q, k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R_{jm}} - 1}{log_2(1+m)'} \tag{2.3}$$

where Z_{kj} is the normalization factor.

Chapter 3

Datasets

Add statistics and examples

Elaborate on splits

Add motivation, gist of each dataset

Add length distribution

3.0.1 Fine-Tuning

As discussed in Section 1, applying BERT to document retrieval requires leveraging passage- or sentence-level relevance judgements fortuitously available in large text collections. Since no such newswire collection currently exists, we train the BERT relevance classifier on three out-of-domain collections.

TREC Microblog (MB)

TREC Microblog datasets draw from the Microblog Tracks at TREC from 2011 to 2014, with topics (i.e., queries) and relevance judgments over tweets. We use the data prepared by Rao et al. [42]. We extract the queries, tweets and relevance judgements from the dataset, excluding metadata such as query time and URLs of the tweets. Relevance judgements in Microblog are in fact reported in a three-point scale where ("irrelevant", "relevant" and "highly relevant"); however, for the purposes of this work we treat both degrees of relevance as equal. We tokenize both query and tweets...

Type	Training Set	Validation Set	Total
Number of queries	166	59	225
Number of tweets	asd	asd	asd
Percentage of relevant tweets	asd	asd	asd

Table 3.1: ...

We sample 25% of the data for the validation set, and use the rest for fine-tuning BERT. What else?

Add total

MicroSoft MAchine Reading Comprehension (MS MARCO)

MS MARCO [5] features user queries sampled from Bing's search logs and passages extracted from web documents. The dataset is composed of tuples of a query with relevant and non-relevant passages. Each query is associated with sparse relevance judgments by human editors. In other words, some queries may have no relevant passage. Because the dataset was initially built from top 10 passages from the Bing search engine and then annotated, some relevant passages might not have been retrieved with BM25. On average, however, each query is associated with one relevant passage.

The models in Section ref were trained on less than 2% of the total training set due to the size of the dataset and time required to train on it even on TPUs. According to Nogueira et al. [33], training for up to 12.5% of the total data does not improve MRR@10 on the validation set.

TODO

Query: bbc world service staff cuts
Text: irish times : bbc world service confirms cuts : the bbc world service will shed around 650 jobs or more
Relevance: 1 ("relevant")

Figure 3.1: An example of... MB

TREC Complex Answer Retrieval (CAR)

TREC CAR [14] uses queries and paragraphs extracted from English Wikipedia: each query is formed by concatenating an article title and a section heading, and passages in that section are considered relevant. This makes CAR, essentially, a synthetic dataset. ...

TODO

3.0.2 Evaluation

We conduct end-to-end document ranking experiments on three TREC newswire collections: the Robust Track from 2004 (Robust04) and the Common Core Tracks from 2017 and 2018 (Core17 and Core18).

Type	Training Set	Validation Set	Total
Number of queries	12.8M	asd	asd
Number of ?	asd	asd	asd
Percentage of relevant passages	asd	asd	asd

Table 3.2: ...

Query: bbc world service staff cuts
Text: irish times : bbc world service confirms cuts : the bbc world service will shed around 650 jobs or more
Relevance: 1 (“relevant”)

Figure 3.2: An example of... MS MARCO

Query: bbc world service staff cuts
Text: irish times : bbc world service confirms cuts : the bbc world service will shed around 650 jobs or more
Relevance: 1 (“relevant”)

Figure 3.3: An example of... CAR

Robust04

Robust04 comprises 250 topics, with relevance judgments on a collection of 500K documents (TREC Disks 4 and 5 except the Congressional Record). We apply five-fold cross-validation [] on...

Core17 & Core18

Core17 and Core18 have only 50 topics each; the former uses 1.8M articles from the New York Times Annotated Corpus while the latter uses around 600K articles from the TREC Washington Post Corpus. Motivation for this dataset? Similar to Robust04, we split the 50 topics into ...

Type	**Training Set**	**Validation Set**	**Total**
Number of queries	12.8M	asd	asd
Number of ?	asd	asd	asd
Percentage of relevant passages	asd	asd	asd

Table 3.3: ...

Chapter 4

Cross-Domain Relevance Transfer with BERT

Need some images to fill it up...

4.1 Architecture

General description? before or after? The architecture of our proposed system is shown in Figure 4.1, which is composed of a two-stage pipeline where Anserini is responsible for

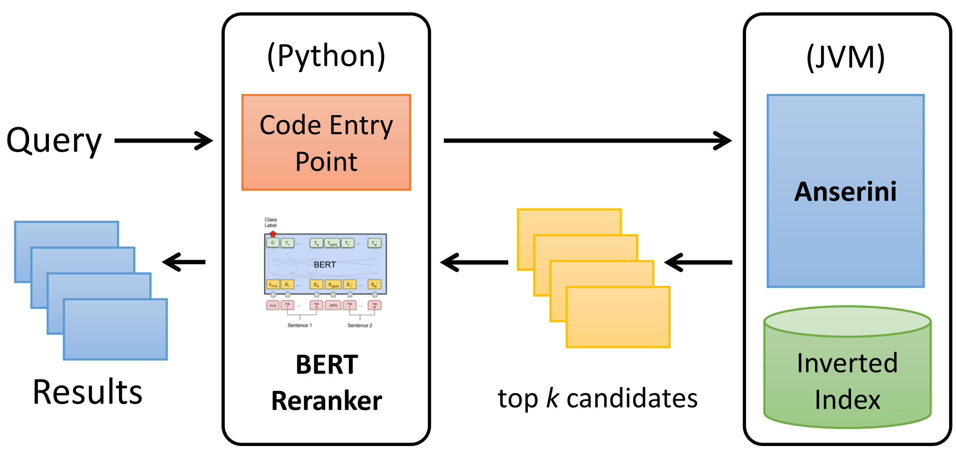

Figure 4.1: ...

retrieval, the output of which is passed to a BERT-based reranker. Our deep learning framework of choice is PyTorch backend? in Python. Nice transition...

4.1.1 Anserini

Anserini architecture, motivation...

Rationale for integration choices, demo stuff...

Reproducibility?

4.1.2 Integration

Smoother integration We apply BERT to document retrieval via integration with an open-source Anserini information retrieval toolkit. Integration of NLP and IR capabilities with distinctly different infrastructures poses a number of technical challenges and requires certain design choices to be considered. 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 [4]. Anserini explanation here? Lucene indexes? 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 that needs to be address for an effective integration.

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[1] 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 4.1), connecting to the backend JVM to access retrieval capabilities.

4.2 Model

How do we solve challenges? Innovations? Add a lot more here?

Go into details of BERT, then how we use it...

Maybe give example for how BERT for relevance modeling looks like? Input? May draw my own diagram

Can I add some math-y stuff to describe what BERT is doing?

The core of our model is a BERT *sentence-level* relevance classifier. What does that mean? Following Nogueira et al. [33], this is framed as a binary classification task. Details!

4.2.1 Input Representation

We form the input to BERT by concatenating the query Q and a sentence S into the sequence [`[CLS]`, Q, `[SEP]`, S, `[SEP]`] and padding each sequence in a mini-batch to the maximum length in the batch. What is Q and S for each dataset? We feed the final hidden state corresponding to the `[CLS]` token in the model to a single layer neural network whose output represents the probability that sentence S is relevant to the query Q. Let's create a half page paragraph here What is the BERT output? What does it mean? Single layer neural network: MLP?

Motivation? Want to combine relevance matching signals To determine *document* relevance, we apply inference over each individual sentence in a candidate document, and then combine the top n scores with the original document score as follows:

$$S_f = a \cdot S_{doc} + (1 - \alpha) \cdot \sum_{i=1}^{n} w_i \cdot S_i \tag{4.1}$$

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

where S_{doc} is the original document score and S_i is the i-th top scoring sentence according to BERT. In other words, the relevance score of a document comes from the combination of a document-level term-matching score and evidence contributions from the top sentences in the document as determined by the BERT model. The parameters α and the w_i's can be tuned via cross-validation.

Go over everything to see if anything needs clarification...

4.3 Experimental Setup

4.3.1 Training and Inference with BERT

We fine-tune $\text{BERT}_{\text{Large}}$ [13] on the datasets discussed in Section **??**. In our implementation we adopt the respective model's `BertForNextSentencePrediction` interface from the Huggingface `pytorch-transformers` (previously known as `pytorch-pretrained-bert`) library[2] as our base model. The maximum sequence length, i.e: 512 tokens, is used for BERT in all our experiments. We train all models using cross-entropy loss for 5 epochs with a batch size of 16. We use Adam [23] with an initial learning rate of 1×10^{-5}, linear learning rate warmup at a rate of 0.1 and decay of 0.1. We conduct all our experiments on NVIDIA Tesla P40 GPUs with PyTorch v1.2.0.

4.3.2 Evaluation

Elaborate? We retrieve an initial ranking of 1000 documents for each query in Robust04, Core17 and Core18 using the open-source Anserini information retrieval toolkit (commit id) based on Lucene 8. To ensure fairness across all three collections, we use BM25 with RM3 query expansion with default parameters. Before running inference with BERT to obtain relevance scores, we preprocess the retrieved documents. First, we clean the documents by stripping any HTML/XML tags and split each document into its constituent sentences witih NLTK. If the length of a sentence with the meta-tokens still exceeds the maximum input length of BERT, we further segment the spans into fixed sized chunks. CV?

Based on preliminary exploration, we consider up to the top three sentences; any more does not appear to yield better results. For Robust04, we follow the five-fold cross-validation settings in Lin et al. [24] and Yang et al. [53]; for Core17 and Core18 we similarly

[2]https://github.com/huggingface/pytorch-transformers

apply five-fold cross validation. The parameters α and the w_i's are learned via exhaustive grid search as follows: we fix $w_1 = 1$ and then vary $a, w_2, w_3 \in [0, 1]$ with a step size 0.1, selecting the parameters that yield the highest average precision (AP). Retrieval results are reported in terms of AP, precision at rank 20 (P@20), and NDCG@20. We report retrieval effectiveness in terms of AP, P@20 and NDCG@20.

Chapter 5

Experimental Results

5.1 Results

Think of nice visualizations... Also add the attention part?

Split into 3 wrt dataset? Also divide into multiple chapters maybe?

The ranking effectiveness for various models on Robust04, Core17 and Core18 are displayed in Table ??. The top row represents the BM25+RM3 query expansion baseline using default Anserini parameters.[1] The remaining blocks belong to the main experiments that we conducted. For instance, MSMARCO → MB refers to a $\text{BERT}_{\text{Large}}$ model first fine-tuned on MS MARCO and then on MB. The nS preceding the model name indicates that inference was performed using the top n scoring sentences from each document. Table ?? also highlights statistically significant results based on paired t-tests compared to the BM25+RM3 baseline with †. We report significance at the $p < 0.01$ level, with appropriate Bonferroni corrections for multiple hypothesis testing.

5.2 Effect of Nature of Training Data

We find that BERT fine-tuned on MB alone significantly outperforms the BM25+RM3 baseline for all three metrics on Robust04. On Core17 and Core18, we observe significant increases in AP as well (and other metrics in some cases). In other words, relevance models

[1] Not tuned for fairness because Core17/18...

	Robust04		
BM25+RM3	0.2903	0.3821	0.4407
1S: $\text{BERT}_{\text{Large}}$(MB)	$0.3318^{\dagger}$	$0.4185^{\dagger}$	$0.4751^{\dagger}$
2S: $\text{BERT}_{\text{Large}}$(MB)	$0.3328^{\dagger}$	$0.4193^{\dagger}$	$0.4765^{\dagger}$
3S: $\text{BERT}_{\text{Large}}$(MB)	$0.3328^{\dagger}$	$0.4193^{\dagger}$	$0.4765^{\dagger}$
1S: $\text{BERT}_{\text{Large}}$(CAR)	$0.3030^{\dagger}$	$0.3980^{\dagger}$	0.4520
2S: $\text{BERT}_{\text{Large}}$(CAR)	$0.3030^{\dagger}$	$0.3980^{\dagger}$	0.4520
3S: $\text{BERT}_{\text{Large}}$(CAR)	$0.3014^{\dagger}$	$0.3964^{\dagger}$	0.4502
1S: $\text{BERT}_{\text{Large}}$(MS MARCO)	$0.3300^{\dagger}$	$0.4309^{\dagger}$	$0.4906^{\dagger}$
2S: $\text{BERT}_{\text{Large}}$(MS MARCO)	$0.3300^{\dagger}$	$0.4339^{\dagger}$	$0.4928^{\dagger}$
3S: $\text{BERT}_{\text{Large}}$(MS MARCO)	$0.3315^{\dagger}$	$0.4321^{\dagger}$	$0.4952^{\dagger}$
1S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	$0.3406^{\dagger}$	$0.4333^{\dagger}$	$0.4945^{\dagger}$
2S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	$0.3436^{\dagger}$	$0.4382^{\dagger}$	$0.5004^{\dagger}$
3S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	$0.3437^{\dagger}$	$0.4390^{\dagger}$	$0.5016^{\dagger}$
1S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB)	$0.3532^{\dagger}$	$0.4462^{\dagger}$	$0.5066^{\dagger}$
2S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB)	$0.3609^{\dagger}$	$\mathbf{0.4624}^{\dagger}$	$0.5231^{\dagger}$
3S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB)	$\mathbf{0.3623}^{\dagger}$	$0.4612^{\dagger}$	$\mathbf{0.5267}^{\dagger}$

Table 5.1: Ranking effectiveness on Robust04

learned from tweets successfully transfer over to news articles despite large differences in domain. This surprising result highlights the relevance matching power introduced by the deep semantic information learned by BERT.

Fine-tuning on MS MARCO or CAR alone yields at most minor gains over the baselines across all collections, and in some cases actually hurts effectiveness. Furthermore, the number of sentences considered for final score aggregation does not seem to affect effectiveness. It also does not appear that the synthetic nature of CAR data helps much for relevance modeling on newswire collections. Interestingly, though, if we fine-tune on CAR and then MB (CAR $\rightarrow$ MB), we obtain better results than fine-tuning on either

		Core17	
BM25+RM3	0.2682	0.5330	0.4329
1S: $\text{BERT}_{\text{Large}}$(MB)	0.2827†	0.5440	0.4443
2S: $\text{BERT}_{\text{Large}}$(MB)	0.2838†	0.5520	0.4526
3S: $\text{BERT}_{\text{Large}}$(MB)	0.2841†	0.5450	0.4489
1S: $\text{BERT}_{\text{Large}}$(CAR)	0.2679	0.5350	0.4338
2S: $\text{BERT}_{\text{Large}}$(CAR)	0.2630†	0.5200†	0.4262
3S: $\text{BERT}_{\text{Large}}$(CAR)	0.2607†	0.5190†	0.4243
1S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3300†	0.4309†	0.4906†
2S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3300†	0.4339†	0.4928†
3S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3315†	0.4321†	0.4952†
1S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	0.2883†	0.5570	0.4559†
2S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	0.2919†	0.5660†	0.4675†
3S: $\text{BERT}_{\text{Large}}$(CAR $\rightarrow$ MB)	0.2926†	0.5660†	0.4685†
1S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB)	0.3091†	0.5710†	0.4770†
2S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB)	0.3175†	**0.5920**†	0.4947†
3S: $\text{BERT}_{\text{Large}}$(MS MARCO $\rightarrow$ MB) **0.3193**†	0.5900†	**0.4998**†	

Table 5.2: Ranking effectiveness on Core17

MS MARCO or CAR alone. In some cases, we slightly improve over fine-tuning on MB alone. One possible explanation could be that CAR has an effect similar to language model pre-training; it alone cannot directly help the downstream document retrieval task, but it provides a better representation that can benefit from MB fine-tuning.

However, we were surprised by the MS MARCO results: since the dataset captures a search task and the web passages are "closer" to our newswire collections than MB in terms of domain, we would have expected relevance transfer to be more effective. Results show, however, that fine-tuning on MS MARCO alone is far less effective than fine-tuning on MB alone.

Looking across all fine-tuning configurations, we see that the top-scoring sentence of each candidate document alone seems to be a good indicator of document relevance, cor-

		Core18	
BM25+RM3	0.3147	0.4720	0.4610
1S: $\text{BERT}_{\text{Large}}$(MB)	0.3288†	0.4780	0.4678
2S: $\text{BERT}_{\text{Large}}$(MB)	0.3314†	0.4810	0.472
3S: $\text{BERT}_{\text{Large}}$(MB)	0.3318†	0.4800	0.4710
1S: $\text{BERT}_{\text{Large}}$(CAR)	0.3129	0.4670	0.4592
2S: $\text{BERT}_{\text{Large}}$(CAR)	0.3128	0.4660	0.4592
3S: $\text{BERT}_{\text{Large}}$(CAR)	0.3130	0.4680	0.4608
1S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3322†	0.4890	0.4845†
2S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3344†	0.4940	0.4876†
3S: $\text{BERT}_{\text{Large}}$(MS MARCO)	0.3368†	0.4950	0.4878†
1S: $\text{BERT}_{\text{Large}}$(CAR → MB)	0.3413†	0.4860	0.4890†
2S: $\text{BERT}_{\text{Large}}$(CAR → MB)	0.3469†	0.4860	0.4859†
3S: $\text{BERT}_{\text{Large}}$(CAR → MB)	0.3472†	0.4870	0.4906†
1S: $\text{BERT}_{\text{Large}}$(MS MARCO → MB)	0.3465†	0.4890	**0.4949**†
2S: $\text{BERT}_{\text{Large}}$(MS MARCO → MB)	0.3497†	0.4840	0.4883†
3S: $\text{BERT}_{\text{Large}}$(MS MARCO → MB)	**0.3511**†	**0.4980**	0.4939†

Table 5.3: Ranking effectiveness on Core18

roborating the findings of []. Additionally considering the second ranking sentence yields at most a minor gain, and in some cases, adding a third actually causes effectiveness to drop. This is quite a surprising finding, since it suggests that the document ranking problem, at least as traditionally formulated by information retrieval researchers, can be distilled into relevance prediction primarily at the sentence level.

In the final block of the table, we present our best model, with fine-tuning on MS MARCO and then on MB. We confirm that this approach is able to exploit *both* datasets, with a score that is higher than fine-tuning on each dataset alone. Let us provide some broader context for these scores: For Robust04, we report the highest AP score that we are aware of (0.3697). Prior to our work, the meta-analysis by [], which analyzed over 100 papers up until early 2019,[2] put the best neural model at 0.3124 [].[3] Furthermore, our results exceed the previous highest known score of 0.3686, which is a non-neural method based on ensembles [10]. This high water mark has stood unchallenged for ten years.

More insight...

5.3 Effect of Length

5.4 Comparison

Recently, [?] reported 0.5381 NDCG@20 on Robust04 by integrating contextualized word representations into existing neural ranking models; unfortunately, they did not report AP results. Our best NDCG@20 on Robust04 (0.5325) approaches their results even though we optimize for AP. Finally, note that since we are only using Robust04 data for learning the document and sentence weights in Eq (4.1), and not for fine-tuning BERT itself, it is less likely that we are overfitting.

Our best model also achieves a higher AP on Core17 than the best TREC submission that does not make use of past labels or human intervention (`umass_baselnrm`, 0.275 AP) [?]. Under similar conditions, we beat every TREC submission in Core18 as well (with the best run being `uwmrg`, 0.276 AP) [2]. Core17 and Core18 are relatively new and thus have yet to receive much attention from researchers, but to our knowledge, these figures represent the state of the art.

[2] https://github.com/lintool/robust04-analysis

[3] Setting aside our own previous work [55].

Chapter 6

Conclusion

References

[1] James Allan, Donna Harman, Evangelos Kanoulas, Dan Li, and Christophe Van Gysel. Trec 2017 common core track overview.

[2] James Allan, Donna Harman, Evangelos Kanoulas, and Ellen Voorhees. TREC 2018 Common Core Track overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference (TREC 2018)*, Gaithersburg, Maryland, 2018.

[3] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

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

[5] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268v3*, 2018.

[6] Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*, 2017.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[8] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.

[9] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[10] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 758–759, New York, NY, USA, 2009. ACM.

[11] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 126–134. ACM, 2018.

[12] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

[14] Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. TREC Complex Answer Retrieval overview. In *Proceedings of the Twenty-Sixth Text REtrieval Conference (TREC 2017)*, Gaithersburg, Maryland, 2017.

[15] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 795–798. ACM, 2015.

[16] Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. Colorless green recurrent networks dream hierarchically. *CoRR*, abs/1803.11138, 2018.

[17] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64. ACM, 2016.

[18] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. *CoRR*, abs/1711.08611, 2017.

[19] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. Neural vector spaces for unsupervised information retrieval. *ACM Trans. Inf. Syst.*, 36(4):38:1–38:25, June 2018.

[20] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[21] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[22] K Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6):809–840, 2000.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[24] Jimmy Lin. The neural hype and comparisons against weak baselines. In *ACM SIGIR Forum*, volume 52, pages 40–51. ACM, 2019.

[25] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. Assessing the ability of lstms to learn syntax-sensitive dependencies. *CoRR*, abs/1611.01368, 2016.

[26] Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship. *arXiv preprint arXiv:1807.03341*, 2018.

[27] Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. CEDR: contextualized embeddings for document ranking. *CoRR*, abs/1904.07094, 2019.

[28] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.

[29] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[31] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*, 2017.

[32] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299. International World Wide Web Conferences Steering Committee, 2017.

[33] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv:1901.04085*, 2019.

[34] Harshith Padigela, Hamed Zamani, and W. Bruce Croft. Investigating the successes and failures of BERT for passage re-ranking. *arXiv:1905.01758*, 2019.

[35] Harshith Padigela, Hamed Zamani, and W. Bruce Croft. Investigating the successes and failures of BERT for passage re-ranking. *CoRR*, abs/1905.01758, 2019.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[37] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[38] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the behaviors of BERT in ranking. *arXiv:1904.07531*, 2019.

[39] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[40] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

[41] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[42] Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Türe, and Jimmy Lin. Multi-perspective relevance matching with hierarchical convnets for social media search. *CoRR*, abs/1805.08159, 2018.

[43] D Sculley, Jasper Snoek, Alex Wiltschko, and Ali Rahimi. Winner's curse? on pace, progress, and empirical rigor. 2018.

[44] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM, 2014.

[45] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[46] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[48] Ellen M. Voorhees. Overview of the TREC 2004 Robust Track. In *TREC 2004*, pages 52–69, 2004.

[49] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[50] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[51] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. *CoRR*, abs/1706.06613, 2017.

[52] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, 10(4):16:1–16:20, October 2018.

[53] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 1129–1132, Paris, France, 2019.

[54] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[55] Wei Yang, Haotian Zhang, and Jimmy Lin. Simple applications of bert for ad hoc document retrieval. *arXiv preprint arXiv:1903.10972*, 2019.

[56] Hamed Zamani and W. Bruce Croft. Relevance-based word embedding. *CoRR*, abs/1705.03556, 2017.

[57] Hamed Zamani, Mostafa Dehghani, W Bruce Croft, Erik Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 497–506. ACM, 2018.

[58] Haotian Zhang, Mustafa Abualsaud, Nimesh Ghelani, Mark D Smucker, Gordon V Cormack, and Maura R Grossman. Effective user interaction for high-recall retrieval: Less is more. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 187–196. ACM, 2018.

[59] Le Zhao and Jamie Callan. Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 259–268. ACM, 2010.

[60] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27, 2015.