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.

Acknowledgements

I would like to thank all the little people who made this thesis possible.

Dedication

This is dedicated to the one I love.

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. [68] 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 [15], 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 [36], [43]. 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 [44], GPT-2 [47]). 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) [16]. 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. [20] 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. [64] 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. [39], 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 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.

1.1.1 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:

$$AP = \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{1.1}$$

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:

$$MAP = \frac{\sum_{j=1}^{|Q|} AP}{Q} = \frac{1}{Q} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{1.2}$$

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 [34]. It is hence one of the standard metrics among the TREC community.

1.1.2 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.

1.1.3 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:

$$NDCG(Q,k) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R_{jm}} - 1}{log_2(1+m)'} \tag{1.3}$$

where Z_{kj} is the normalization factor.

1.2 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 [13] with a non-neural method based on ensembles, which has stood unchallenged for ten years.
- We explore through various analyses the effects of cross-domain relevance transfer with BERT as well as the contributions of BM25 and sentence scores to the final document ranking. We investigate the effect of query and document length on retrieval effectiveness with BM25 and BERT, and the reasons behind the substantial improvements introduced with BERT. Revisit after completing experimental results
- 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.

1.3 Thesis Organization

Update The remainder of this thesis is organized in the following order: Chapter 2 reviews related work in document retrieval, pretrained language models and document re-ranking models based on machine learning techniques. 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

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

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.

Chapter 2

Background and Related Work

2.1 Document Retrieval

doc retrieval architectures - multi-stage retrieval, bag of words, rerankers 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 [26]. 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 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 [29].

2.2 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 [30] and hierarchical patterns [19]. In general, pretrained language models can be applied to downstream tasks in one of two ways: feature-based and fine-tuning.

2.2.1 Feature-based Approaches

The feature-based approach, such as ELMo [44], 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 [44] 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 [35] and meaning while the lower levels learn syntactic features well [6]. While traditional pretrained word embeddings like GloVe [43] 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. [44] 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 [48] and sentiment analysis on the Stanford Sentiment Treebank (SST-5) [52].

2.2.2 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. [47] 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 [56] with impressive results. At the core of GPT-2 lies a multi-layer left-to-right transformer [54] decoder, with each layer consisting of a multi-head self-attention mechanism and fully connected feed-forward network [46]. The large capacity of the transformer is exploited by pretraining it on Google BookCorpus dataset [69] (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 [47], Bidirectional Encoder Representations from Transformers (BERT) [16] 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. [16] 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 [54]: 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 [69] (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 with WordPiece embeddings [57] 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 soft-

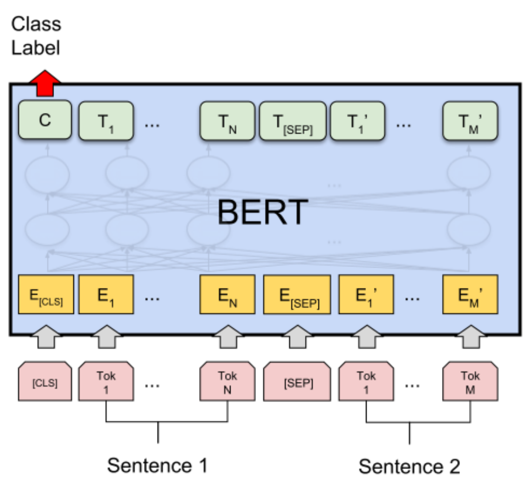

Figure 2.1: The architecture combining BERT an additional output layer for the sentence pair classification task, where E represents the input embedding for each sentence and T_i the contextual representation of token i. Adapted from Delvin et al. [16].

max. 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. [63] which integrated BERT with Anserini for question answering over Wikipedia by fine-tuning BERT on SQuAD, and Nogueira et al. [39] who adopted BERT for passage reranking over MS MARCO.

2.3 Machine-Learned Ranking Models

Come up with a better title

2.3.1 Learning to Rank Methods

Learning to rank (LTR) methods have arisen in document retrieval to apply supervised machine learning techniques to automatically learn a ranking model. This trend has started in response to a growing number of relevance signals, particularly in web search, such as anchor texts. In this setup, training samples are extracted from large amounts of search log data in the form of a list of documents and their relevance labels for a number of queries. Unlike traditional ranking models, LTR models require a good deal of feature engineering, which can be time-consuming and difficult to generalize.

Existing algorithms for LTR can be categorized into three categories based on their input representation and loss function: pointwise, pairwise or listwise. [32] Pointwise algorithms such as McRank [28] approximate the LTR problem as a regression problem where a relevance score is predicted for each query-document pair. On the other hand, LTR is framed as a classification problem by pairwise algorithms like RankSVM [25] and LambdaMART [10] to choose the more relevant one given a pair of documents. Listwise algorithms such as ListNet [11] instead optimize the relevance score over the entire list of documents for a query. [32] claim that listwise LTR approaches often produce better rankings in practice.

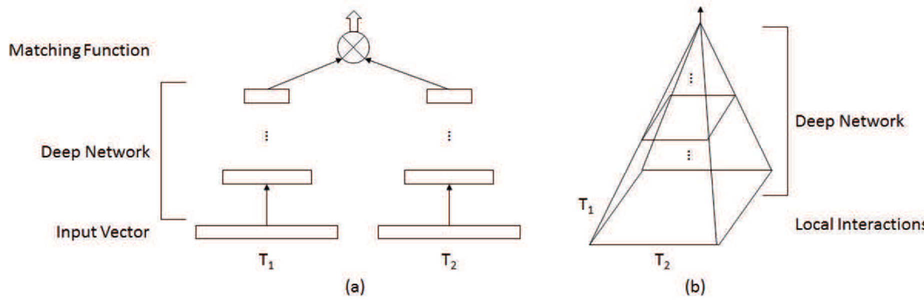

Figure 2.2: The two types of deep matching architectures: representation-focused (a) and interaction-focused (b). Adapted from Guo et al. [21].

2.3.2 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 [36] trained on Google News, GloVe [43] 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 [53]. There has also been some effort in learning word embeddings to directly capture relevance matching [65, 18] rather than linguistic features as in word2vec [36] or GlovE [43]. 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) [23] 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 [51] 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. [51] 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 [21], KNRM [58] and DUET [38].

DRMM [21], 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 [58] 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 [38] 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 2.3.2 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 2.2 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 2.1.

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 2.2. 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. [39] 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. [41] 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 [25] and a number of neural kernel matching models such as KNRM [58] and Conv-KNRM [14], 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. [64] 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. [67] 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. [64] pave the way for future work that culminated in this thesis.

More recently, MacAvaney et al. [33] 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) [33] with KNRM [58]. 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. [45] 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. [39] 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 [58] and Conv-KNRM [14]. 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.

2.3.3 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 [29], 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 [29] also echoes the general skepticism concerning the empirical rigor and contributions of machine learning applications in Lipton et al. [31] and Sculley at al. [50]. In particular, Lin et al. [29] 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. [62] 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 [61] 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

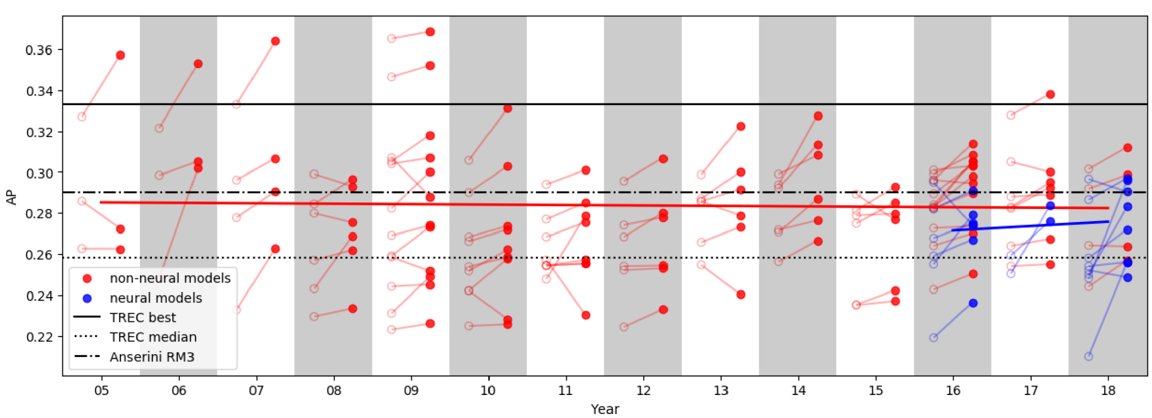

Figure 2.3: Visualization of best AP scores on Robust04 for 108 papers based on non-neural and neural approaches. Adapted from Yang et al. [62].

report AP scores higher than the TREC best, with the highest being by Cormack et al. [13] 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. [66] in 2018 at AP 0.2971. Deviating from the dominant approach of deploying neural models as rerankers, Zamani et al. [66] 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. [13]. 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. [29]. 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. [22] 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 methods on both Core17 and Core18 has been limited.

Chapter 3

Cross-Domain Relevance Transfer with BERT

Our proposed model is based on sentence-level relevance modeling and document re-ranking with BERT. By training a BERT-based relevance classifier, we aim to extract valuable semantic matching signals which can be leveraged to re-rank a list of candidate documents retrieved with BM25 + RM3. We also explore applying cross-domain relevance transfer to exploit models of relevance learned on out-of-domain collections. This approach is crucial in re-ranking documents that are too long for BERT to handle. This chapter describes the details of our BERT-based sentence-level relevance classifier and document re-ranker.

3.1 Datasets

Move this elsewhere? Add length distribution?

In order to model sentence-level relevance with BERT, we need training pairs of short text and relevance judgements. A number of collections fortuitously contain relevance judgements at the sentence and passage level, which makes them the ideal choice for fine-tuning BERT. We fine-tune BERT on three such sentence- and passage-level datasets individually and in combination: TREC Microblog, MicroSoft MAchine Reading Comprehension and TREC Complex Answer Retrieval datasets. The details of each dataset are provided below.

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: TODO

3.1.1 TREC Microblog (MB)

The TREC Microblog dataset draws from the Microblog Tracks at TREC from 2011 to 2014, with topics and relevance judgments over tweets. Topics associated with tweets are treated as queries, and each of the four datasets contains 50 queries. The nature of this collection differs from newswire documents that we evaluate our models on in distinct ways: First of all, tweets have much fewer tokens than newswire documents. By definition, tweets are limited to 280 characters. The length distribution of tweets in MB is displayed in Figure X. Furthermore, because queries and tweets in this dataset are comparable in length, exact matches of query terms occur less frequently in the tweets than they might in longer documents such as news articles. Therefore, semantic matching signals may take precedence in improving retrieval effectiveness on MB. How is this relevant to our training? It's valuable because... Related to this point, tweets are expressed in a much less formal language than news articles. Tweets may characteristically contain various abbreviations (partly due to the aforementioned length constraint), informal conventions such as hashtags or typos. Such informal language may result in term mismatches in the case of exact matching. It may therefore be helpful to catch other semantic signals with a deep neural network.

We use the MB data prepared by Rao et al. [49]. 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 MB are in fact reported on a three-point scale

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: TODO

where ("irrelevant", "relevant" and "highly relevant"); however, for the purposes of this work we treat both higher degrees of relevance as equal [40]. Both queries and tweets are segmented into token sequences with the Stanford Tokenizer Tool[1]. We sample 25% of the data for the validation set, and use the rest for fine-tuning BERT. We experiment with different splits as discussed in Section Exp-results, and find this split to be ideal.

3.1.2 MicroSoft MAchine Reading Comprehension (MS MARCO)

MS MARCO is a large-scale machine reading comprehension and question answering dataset that is extensively used in the NLP community. 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. On average, each query has one relevant passage. However, some may have no relevant passage at all as the dataset is constructed from top-10 passages manually annotated by human judges. Therefore, some relevant passages might not have been retrieved with BM25. Rephrase this MS MARCO can be distinguished from similar datasets by its size and real-world nature. Similar to MB, MS MARCO is representative of a natural, and noisy, distribution of information needs, unlike other datasets that often contain high-quality text that may not reflect the use in real life. What else? Robust systems

[1]https://nlp.stanford.edu/software/tokenizer.shtml

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: TODO

Query: bbc world service staff cuts
Relevant Passage: irish times : bbc world service confirms cuts : the bbc world service will shed around 650 jobs or more
Non-relevant Passage: irish times : bbc world service confirms cuts : the bbc world service will shed around 650 jobs or more

Figure 3.2: TODO

Here we focus on the passage-ranking task on MS MARCO. Following the settings in Nogueira et al. [], we train BERT on approximately 400M training samples. The development set is composed of approximately 6.9k queries, each paired with the top 1000 most relevant passages in the MS MARCO dataset as retrieved with BM25. Similarly, the evaluation set contains approximately 6.8 queries and their top 1000 passages, but without the relevance annotations. The models in Section X were trained on less than 2% of the total training set (12.8M) due to the size of the dataset and time required to train on it even on TPUs. According to Nogueira et al. [39], training for up to 12.5% of the total data does not improve MRR@10 on the validation set. Maybe remove MRR, and better transition

3.1.3 TREC Complex Answer Retrieval (CAR)

TREC CAR [17] uses paragraphs extracted from all paragraphs in the English Wikipedia, except the abstracts. How many queries? Each query is formed by concatenating an article title and a section heading, with all passages under that section considered relevant. The organizers of TREC CAR 2017 only provide manual annotations for the top-5 passages retrieved, meaning some relevant passages may not be annotated if they rank lower. For this reason, we opt to use automatic annotations that provide relevance judgements for all possible query-passage pairs. The goal of this TREC track is to automatically collect and condense information for a complex query into a single coherent summary. Rather than focusing on document retrieval, the priority is aggregating synthesized information in the form of references, facts and opinions. However, CAR is a synthetic dataset in the sense that queries and documents do not reflect real-world distributions or information needs. More?

The dataset has five predefined folds over the X queries. Paragraphs corresponding to the first four folds are used to construct the training set (consisting of approximately 3M queries), and the rest the validation set (approximately 700K queries). The original test set used to evaluate submissions to TREC CAR is used for testing purposes (approximately 1.8k queries). A subtle detail to note is that rhe official BERT models are pre-trained on the entire Wikipedia dump; therefore, they have also been trained on documents in the TREC CAR test collection albeit in an unsupervised fashion. In order to avoid the leak of test data into training, we use the BERT model pre-trained only on the half of Wikipedia present in CAR training samples []. 30M fine-tuning query-passage pairs were generated by retrieving the top 10 passages from the entire CAR corpus with BM25. Similar to MS MARCO, training on more than 40% of the training set did not lead to any improvements on the validation set. cite rodrigo?

3.2 Modeling Relevance with BERT

We propose modeling sentence-level and passage-level relevance with BERT to capture semantic signals. Better alternative to capture semantic signals? This approach is motivated by the application of transfer learning in NLP where a large transformer model trained for language modeling can be used for various downstream tasks. Add one more sentence to make this better Specifically, BERT is trained on copious amounts of unsupervised data from the Google BookCorpus and English Wikipedia with masked language modeling. Talk about relevance prediction too Although the training procedure doesn't involve any explicit objective to extract linguistic features, it has been shown to implicitly recognize such features as subject-verb agreement and conference resolution [24, 12]. More on this? This phenomenon allows a number of NLP tasks to greatly benefit from featured implicitly encoded in BERT weights.

3.2.1 Relevance Classifier

The core of our model is a BERT-based sentence-level relevance classifier. In other words, we aim to build a model on top of BERT to predict a relevance score s_i for a sentence or passage d_i given a query q. Because the maximum input length that BERT can handle is 512 tokens, we limit our training data to sentence- and passage-level datasets as explained in Section X. In other words, d_i are either tweets drawn from TREC Microblog or passages

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: 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 3.3: TODO

blahblahblah

Figure 3.4: Put tokenized BERT input example?

from MS MARCO or TREC CAR. Following Nogueira et al. [39], we frame relevance modeling as a binary classification task. Figure X illustrates how BERT can be used for to predict the relevance of a given sentence. Add diagram to explain the relevance modeling process More specifically, we feed query-text pairs into the BERT model with their respective relevance judgements (i.e: 0 for non-relevant and 1 for relevant). Through this training process BERT learns to automatically judge whether a piece of unseen text is relevant for a given query or not. The details of the input representation to BERT is discussed at length in the next section. The specifics of fine-tuning BERT for relevance classification is also explained in Section X.

Input Representation

We form the input to BERT by concatenating the query q and a sentence d into the sequence [`[CLS]`, q, `[SEP]`, d, `[SEP]`] . The `[SEP]` metatoken is used to distinguish between two non-consecutive token sequences in the input, i.e: query and text, and the `[CLS]` signifies a special symbol for classification output. Although BERT supports variable length token sequences, the final input length must be consistent across training. Therefore, we pad each sequence in a mini-batch to the maximum length in the batch.

The complete input embeddings of BERT is comprised of token, segmentation and

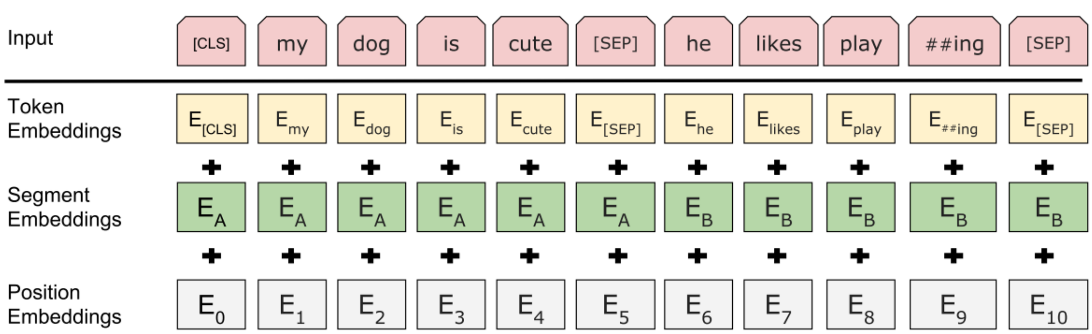

Figure 3.5: Create customized diagram

position embeddings. The first is constructed by tokenizing the above sequence with the proper metatokens in place with the BERT tokenizer. Since BERT was trained based on WordPiece tokenization, we use the same tokenizer to achieve optimal performance. WordPiece tokenization may break words into multiple subwords in order to more efficiently deal with out-of-vocabulary words and better represent complex words. During training, the subwords derived with WordPiece tokenization are reconstructed based on the training corpus. After tokenization, each token in the input sequence is converted into token IDs corresponding to the index in BERT's vocabulary. Tokens that do not exist in the vocabulary are represented with a special `[UNK]` token.

The segment embeddings indicate the start and end of each sequence, whether it be a single sequence or a pair. For relevance classification where we have two texts in the input sequence, i.e: query and sentence, the segment embeddings corresponding to the tokens of the first sequence, i.e: the query, are all 0's, and those for the second sequence, i.e: the document, are all 1's. The position embeddings are learned for sequences up to 512 tokens, and help BERT recognize the relative position of each token in the sequence. An example BERT input for MB is shown in Figure X. Example with complex words

Fine-Tuning

Go more into detail about NN stuff? Many relevant features such as synonyms and long-term dependencies are already encoded in pretrained BERT weights. It is thus possible to fine-tune BERT with less data and time by adding a fully connected layer on top of the network. Intuitively, the lower layers of the network have already been trained to capture features relevant to the task.

To fine-tune BERT for relevance modeling, we add a single layer neural network on top of BERT for classification. This layer consists of $K \times H$ randomly initialized neurons where K is the number of classifier labels and H is the hidden state size. For relevance classification, we have two labels indicating whether the sentence is relevant or non-relevant for the given query ($K = 2$).

The final hidden state corresponding to the first token, i.e: `[CLS]`, provides a H-dimensional aggregate representation of the input sequence that can be used for classification. We feed the final hidden state in the model to the single layer neural network. The probability that the sentence d is relevant to the query q is thus computed with standard softmax:

$$\sigma(y_j) = \frac{e^{y_j}}{\sum_i e^{y_i}} \tag{3.1}$$

where $\sigma(y_j)$ maps the arbitrary real value y_j into a probability distribution. Intuitively, $\sigma(y_j)$ represents the probability of the relevance of sentence d. The parameters of BERT and the additional softmax layer are optimized jointly to maximize the log-probability of the correct label with cross-entropy loss.

3.3 Reranking with BERT

Fine-tuning BERT on relevance judgements of query-text pairs allows us to obtain a model of relevance so that we can compute sentence-level relevance scores easily on any collection. However, recall that we trained BERT on sentence- or passage-level text so as not to exceed the maximum input size of BERT. These training datasets come from very different distributions than newswire collections as discussed in Section **??**. May need to move dataset stuff out of evaluation? In order to predict relevance on much longer newswire

Figure 3.6: ...

documents, we explore cross-domain relevance transfer by using the same models. Our hypothesis is that if a neural network with a large capacity such as BERT can capture relevance in one domain, it might be able to transfer to other domains. Probably need a couple more sentences to make this more coherent

For this reason, we split each relevant document as retrieved with BM25 + RM3 into its constituent sentences with the Stanford tokenizer. We then run inference over this new sentence-level collection with our fine-tuned models to compute a score for each sentence. We determine overall document scores by combining exact and semantic matching signals. Based on BM25 + RM3 document scores we know a ranking of documents with respect to exact matches. Sentence-level scores obtained with BERT reveal other implicit semantic information not evident to BM25. By combining the two sets of relevance matching signals, we discover a more diverse notion of relevance, leading to a better ranking of documents. Give example diagram of sentence score ranking? Maybe with BM25?

Using either set of scores to rank documents neglects crucial information from the other, so we interpolate the scores. Therefore, to determine overall document relevance, we combine the top n scores with the original document score as follows:

$$S_f = a \cdot S_{doc} + (1 - \alpha) \cdot \sum_{i=1}^{n} w_i \cdot S_i \tag{3.2}$$

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. Motivation?

3.4 Experimental Setup

3.4.1 Training and Inference with BERT

We fine-tune $\text{BERT}_{\text{Large}}$ [16] on the datasets discussed in Section **??**: TREC Microblog, MS MARCO AND TREC CAR. In our implementation we adopt the respective model's `BertForNextSentencePrediction` interface from the Huggingface `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.

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

The fine-tuning procedure introduces few new hyperparameters to those already used in pre-training: batch size, learning rate, and number of training epochs. Due to the large amount of training data in MS MARCO and TREC CAR, BERT is initially trained on Google's TPU's with a batch size of 32 for 400k iterations, following [39]. We use Adam [27] with an initial learning rate of 3×10^{-6}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and L2 decay of 0.01. Learning rate warmup is applied over the first 10k steps with linear decay of learning rate. We apply a dropout probability of 0.1 across all layers.

We train all other models using cross-entropy loss for 5 epochs with a batch size of 16. We conduct all our experiments on NVIDIA Tesla P40 GPUs with PyTorch v1.2.0. We use Adam [27] with an initial learning rate of 1×10^{-5}, linear learning rate warmup at a rate of 0.1 and decay of 0.1. Applying diminishing learning rates is especially important in fine-tuning BERT in order to preserve the information encoded in the original BERT weights and speed up training.

3.4.2 Evaluation

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's Stanford Tokenizer. 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.

Following the procedure in Section X, we obtain a relevance score for each sentence. We experiment with the number of top scoring sentences to consider while computing the overall score, and find that using only the top 3 sentences is often enough. In general, considering any more doesn't yield better results. To tune hyperparameters in Equation X, we apply five-fold cross-validation over TREC topics. For Robust04, we follow the five-fold cross-validation settings in [29] over 250 topics. For Core17 and Core18 we similarly apply five-fold cross validation. We learn parameters α and the w_i on four folds via exhaustive grid search with $w_1 = 1$ and varying $a, w_2, w_3 \in [0, 1]$ with a step size 0.1, selecting the values that yield the highest AP on the remaining fold. We report retrieval effectiveness in terms of AP, P@20, and NDCG@20.

Chapter 4

Architecture

We apply BERT to document retrieval via integration with the open-source information retrieval toolkit Anserini[1]. The proposed architecture of our system follows a two-stage pipeline as shown in Figure 4.1: Anserini is used to retrieve documents from indexed collections with BM25, forming an initial candidate list. Our model introduced earlier in Section 3 is deployed as a re-ranker over the candidate documents based on sentence-level relevance to produce a final ranking of documents. In this section we review each component of the architecture and discuss the design choices behind their integration. We

[1] http://anserini.io/

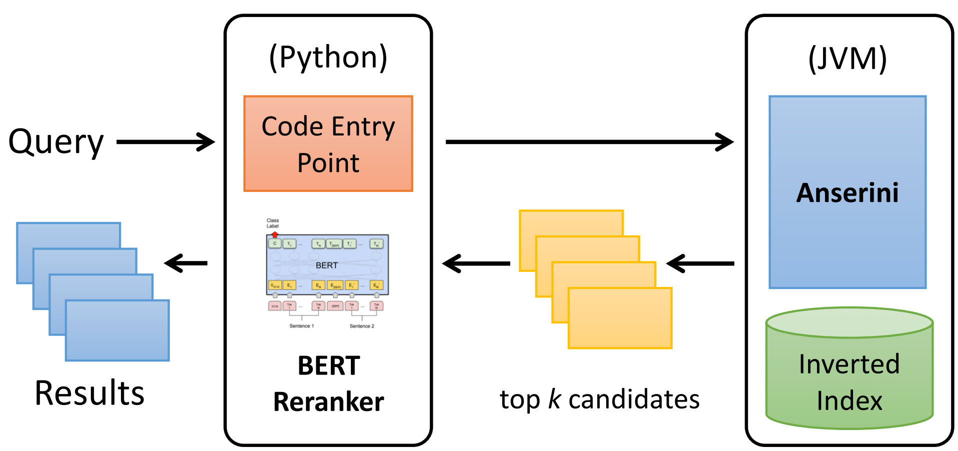

Figure 4.1: Update

also touch upon the issue of reproducibility in information retrieval and our efforts to make our work more reproducible as well as their limitations .

4.1 Anserini

Technology transfer between the academic and industry information retrieval communities is at times impeded due to a lack of universal set of tools and infrastructure. Most industry practitioners have adopted Lucene[2], Solr[3] or Elasticsearch[4] as the de facto platform in the development of search applications with the primary objective of scalability. However, academic systems such as Indri[5] and Terrier[6] are far more prevalent among researchers, which prioritize better rankings above all else with little consideration for operational characteristics.

Anserini [59, 60] was developed in response to this disconnect to provide a research-focused information retrieval toolkit on top of the open-source Lucene search library. Anserini facilitates efficient full text indexing and search capabilities over large-scale text collections by providing wrappers and intuitive APIs on top of core Lucene libraries. More importantly, Anserini makes it possible for researchers and industry practitioners alike to systematically evaluate their models over standard test collections in a reproducible and comparable manner.

Related to our work, Anserini can be seamlessly integrated into multi-stage ranking architectures with large improvements in retrieval effectiveness and low latency. We initially use Anserini to index our test collections in a multi-threaded manner with Lucene 8.0 (post commit id `75e36f9` which has greatly improved query evaluation latency over the previous Lucene 7.6). An initial ranked list of documents is retrieved to depth 1000 for each query with BM25 using default Anserini parameters.

4.2 Python Module

Need a better name that doesn't include Python... The Python module lies at the core of our system, encompassing the preprocessing, training / inference and evaluation compo-

[2]https://lucene.apache.org/
[3]https://lucene.apache.org/solr/
[4]https://www.elastic.co/
[5]https://www.lemurproject.org/
[6]http://terrier.org/

nents. All the functionalities of our proposed model in Section 3 are implemented in this module in Python using the deep learning framework PyTorch.

The preprocessing component consumes the documents retrieved with Anserini to convert them into a format that can be used by the main component that enables training and inference with BERT. On the one hand, the main component can be used to train BERT as a relevance classifier. This functionality may be used independently of the overall pipeline to fine-tune BERT on new collections. On the other hand, we can run inference over the output of the preprocessing module with previously trained models, producing a list of sentence relevance scores. Finally, this component also serves as a re-ranker where sentence and BM25 document scores are interpolated in order to compute an overall relevance score for each candidate document. Last but not least, the evaluation component integrates directly with Anserini to assess the retrieval effectiveness of our system.

4.3 Integration

Our two-stage pipeline marries NLP and IR capabilities to implement an effective scalable document retrieval system that successfully leverages semantic cues. For an effective integration, we need to address the the technical challenge of connecting the two components with different infrastructural requirements. In this section we discuss design choices to bridge the worlds of NLP and IR from a software engineering viewpoint.

In our architecture Anserini, which is responsible for indexing and retrieval in our system, runs on the Java Virtual Machine (JVM) as it is mostly implemented in Java or provides Python wrappers on Java. However, our deep learning framework of choice PyTorch, similar to alternatives such as TensorFlow[7], are implemented in Python with a C++ backend to support X.

There exist two immediate solutions to bridge Python and the JVM. "Loosely-coupled" integration approaches involve using an intermediary medium between Python and the JVM. For example, we may pass text files between the two in order to facilitate communication without direct interaction. However, this is not an efficient solution as it requires writing / reading potentially large files to / from disk, not to mention the memory requirements. Furthermore, this approach requires diligent monitoring to ensure that changing file formats and APIs do not break code. Integration via REST APIs is plagued with similar issues as passing intermediate text files. Specifically, it may require frequent HTTP calls, thus introducing significant overhead. Additionally, imperfect solutions for enforcing API

[7] https://www.tensorflow.org/

contracts risk stability of the system. Ultimately, neither approach is suitable for rapid experimentation in a research environment.

Therefore, we explore ways to achieve "tightly-coupled" integration. One solution is to adopt the Java Virtual Machine (JVM) as the primary code entry point, and connect to PyTorch's C++ backend via the Java Native Interface (JNI). However, this would result in two separate code paths (JVM to C++ for execution and Python to C++ for model development), leading to maintainability issues similar to those mentioned with regard to REST APis.

For this reason, we finally chose Python as our primary development environment, integrating Anserini using the Pyjnius Python library[8] for accessing Java classes. Pyjnius was originally developed to facilitate Android development in Python, and allows Python code to directly manipulate Java classes and objects. Thus, our system supports Python as the main development language (and code entry point, as shown in Figure 4.1), connecting to the JVM to access retrieval capabilities of Anserini.

4.4 Replicability and Reproducibility

Over the last decade, it has become increasingly challenging to verify reported results and compare various performance metrics due to growing number of information retrieval systems both in the academia and the industry. Unlike some fields of computer science where it is practical to manually corroborate findings or visually inspect results, the amount and type of data involved in document retrieval deems this approach infeasible. As a matter of fact, this challenge has prompted one of the largest IR conferences in the world, SIGIR, to issue a task force to determine guidelines to establish repeatability, replicability and reproducibility principles in IR projects.[9]

Repeatability emphasises a researcher's ability to reliably repeat her own computation. The path to this goal is through rigorous logging, good data management practices and consistent use of virtual environments. We don't delve further into the details of repeatability as the practices we follow are universal to X

Replicability highlights the ability of an independent group to obtain the same results using the developer's original artifacts. We strive to make our work replicable by building a Docker image to accompany our system that allows anyone to deploy and test our

[8] `https://pyjnius.readthedocs.io/`

[9] http://sigir.org/wp-content/uploads/2018/07/p004.pdf

system on any operating system easily. By adhering to the requirements defined in the SIGIR Open-Source IR Replicability Challenge (OSIRRC), we ensure that our system can seamlessly work with their evaluation infrastructure in the future. The image is available on Docker hub with the tag `tag`. The OSIRRC jig[10] needs to be set up first to run the commands on Docker hub. The OSIRRC Docker container contract includes three "hooks" for interacting with the system: The `init` hook has to be called first, whose purpose is to run any preparatory steps for the retrieval run including downloading and compiling the source code, downloading pre-built artifacts such as JAR files and other external resources such as pretrained models. In our case, we pull the source code, data and pretrained models from Google Cloud Storage buckets; build Anserini with Maven, and the TREC evaluation tool. Next the `index` hook is called to, as the name indicates, build the necessary indexes. Finally, the `search` hook helps perform multiple ad-hoc retrieval runs in a row. Each of the hook scripts accepts a JSON file that defines the various arguments for the respective script such as path to the relevance judgements file.

Reproducibility refers to the the ability of an independent group of researchers to implement the author's proposed artifacts from scratch with the same results. This final goal is indeed the hardest to achieve; as a matter of fact, it may even be impossible in certain cases due to non-determinism. Unfortunately, we found this to be true with our work with BERT as well. For example, the fine-tuning and inference processes described in Section X produces slightly different sentence scores (in the order of X) unless they are performed on the same GPU. To further aggravate this issue, these small differences add up over floating point operations, leading to as much as a 0.5 point difference in AP. Score ties, hyperparameters?

[10]https://github.com/osirrc/jig

Chapter 5

Experimental Results

5.1 Datasets

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).

Robust04

Robust04 draws from the TREC Robust Track in 2004, which is the set of documents from TREC Disks 4 and 5, spanning news articles from Financial Times and LA Times, except the Congressional Record. The dataset comprises 250 topics, with relevance judgments on a collection of 500K documents. The goal of the Robust track is to improve the consistency and robustness of retrieval methods by focusing search on poorly performing topics . Specifically, this task involves searching across a fixed set of documents using previously unseen topics. Notably the lengths of documents in Robust04 are highly biased: data, which is difficult for neural text matching models to handle. What does this mean for us?

Core17 & Core18

Core17 and Core18 are based on the TREC 2017 and 2018 Common Core Tracks respectively. The motivation behind these tracks is to build up-to-date test collections based on more recently created documents. that avoids the pitfalls of depth-k pooling Core17 uses

Table 5.1: Ranking effectiveness on Robust04, Core17 and Core18

	Robust04			Core17			Core18		
Model	**MAP**	**P@20**	**NDCG@20**	**MAP**	**P@20**	**NDCG@20**	**MAP**	**P@20**	**NDCG@20**
BM25+RM3	0.2903	0.3821	0.4407	0.2823	0.5500	0.4467	0.3135	0.4700	0.4604
1S: BERT(MB)	0.3408†	0.4335†	0.4900†	0.3091†	0.5620	0.4628	0.3393†	0.4930	0.4848†
2S: BERT(MB)	0.3435†	0.4386†	0.4964†	0.3137†	0.5770	0.4781	0.3421†	0.4910	0.4857†
3S: BERT(MB)	0.3434†	0.4422†	0.4998†	0.3154†	0.5880	0.4852†	0.3419†	0.4950†	0.4878†
1S: BERT(CAR)	0.3025†	0.3970†	0.4509	0.2814†	0.5500	0.4470	0.3120	0.4680	0.4586
2S: BERT(CAR)	0.3025†	0.3970†	0.4509	0.2814†	0.5500	0.4470	0.3116	0.4670	0.4585
3S: BERT(CAR)	0.3025†	0.3970†	0.4509	0.2814†	0.5500	0.4470	0.3113	0.4670	0.4584
1S: BERT(MS MARCO)	0.3028†	0.3964†	0.4512	0.2817†	0.5500	0.4468	0.3121	0.4670	0.4594
2S: BERT(MS MARCO)	0.3028†	0.3964†	0.4512	0.2817†	0.5500	0.4468	0.3121	0.4670	0.4594
3S: BERT(MS MARCO)	0.3028†	0.3964†	0.4512	0.2817†	0.5500	0.4468	0.3121	0.4670	0.4594
1S: BERT(CAR → MB)	0.3476†	0.4380†	0.4988†	0.3103†	0.5830	0.4758	0.3385†	0.4860	0.4785
2S: BERT(CAR → MB)	0.3470†	0.4400†	0.5015†	0.3140†	0.5830	0.4817†	0.3386†	0.4810	0.4755
3S: BERT(CAR → MB)	0.3466†	0.4398†	0.5014†	0.3143†	0.5830	0.4807	0.3382†	0.4830	0.4731
1S: BERT(MS MARCO → MB)	0.3676†	0.4610†	0.5239†	0.3292†	0.6080†	0.5061†	0.3486†	**0.4920**	**0.4953**†
2S: BERT(MS MARCO → MB)	**0.3697**†	0.4657†	0.5324†	**0.3323**†	0.6170†	**0.5092**†	0.3496†	0.4830	0.4899†
3S: BERT(MS MARCO → MB)	0.3691†	**0.4669**†	**0.5325**†	0.3314†	**0.6200**†	0.5070†	**0.3522**†	0.4850	0.4899†

1.8M articles from the New York Times Annotated Corpus while Core18 uses around 600K articles from the TREC Washington Post Corpus. Core17 and Core18 have only 50 topics each, which are drawn from the Robust Track topics.

5.2 Results and Analyses

Table 5.1 presents our main results on Robust04, Core17 and Core18. The top row, BM25+RM3, corresponds to the BM25 runs with RM3 query expansion using default Anserini parameters. Although higher scores can be obtained on Robust04 with tuned parameters, we present untuned runs for the sake of fairness as no careful tuning has been performed for Core17 or Core18. The remaining five blocks show the retrieval effectiveness of our models trained as described in Section X. The models are labeled to reflect the fine-tuning procedure where the datasets that a $\text{BERT}_{\text{Large}}$ model was trained on are listed in order in parantheses. The nS preceding the model name indicates that the top n sentences were interpolated to compute an overall document score. Table 5.1 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.

5.2.1 Effect of Training Data

By fine-tuning BERT on three different datasets alone and in combination, we hope to study the effect of nature and amount of training data on the power of our learned relevance matching model. As Table 5.1 shows, the particular source of relevance labels that we train BERT on dramatically influences retrieval effectiveness across all three collections.

First of all, we find that fine-tuning BERT on MB alone, i.e: BERT(MB), significantly outperforms the BM25+RM3 baseline for all metrics on Robust04. We observe significant increases in AP on Core17 and Core18 as well and in P@20 and NDCG@20 in some cases. These results confirm that relevance models learned from tweets can be successfully transferred to news articles in spite of the considerable differences in domain and style. This surprising finding may be attributed to the relevance matching power of BERT. Reword MB portion?

Unlike MB, fine-tuning on MS MARCO or CAR results in marginal gains over the baseline on Robust04. Reranking with these models in fact hurts effectiveness on Core17 and Core18 across all metrics. The synthetic nature of CAR data especially does not appear to be useful for relevance modeling on newswire collections. For instance, BERT(CAR) leads to 0.3120 AP on Core18 compared to the 0.3135 AP of the baseline, which means that the model actually picks irrelevant documents as the most relevant to the query. As the results $2S$ and $3S$ for the same model show, the effectiveness on Core18 in fact progressively degrades the more sentences are considered in final score aggregation. Intuitively these results indicate that using these models disrupt the order imposed by the baseline. For the other experiments with these two models. the number of sentences considered does not seem to affect effectiveness at all.

Results with BERT(MS MARCO) are more surprising in that the MS MARCO dataset captures a search task, and web passages in the dataset are closer to the newswire test collections than MB in terms of domain. Given the proximity of domains, it would be reasonable to expect relevance transfer between MS MARCO and newswire collections to be more effective. However, our results indicate that this is not necessarily the case, and fine-tuning on MS MARCO alone rather than MB alone is far less effective for relevance transfer.

Although fine-tuning on CAR or MS MARCO alone does not yield large improvements, we actually obtain considerably higher results by fine-tuning these models further on MB. With BERT(CAR $\rightarrow$ MB) we achieve effectiveness that is slightly better than fine-tuning on MB alone in some cases. We hypothesize that CAR might have a similar effect to language pre-training in that it doesn't directly apply to the downstream document retrieval task, but provides a better representation that can benefit from fine-tuning on MB. More surprisingly, fine-tuning on MS MARCO first and then on MB represents our best model (BERT(MS MARCO $\rightarrow$ MB)) as shown in the final block of the table. This model is able to exploit data from both MS MARCO and MB, with a score that is higher than fine-tuning on each dataset alone.

5.2.2 Comparison to Other Ranking Models

Focus on ML-based models In this section we try to put our results into perspective by considering them in the larger context of document retrieval research. For Robust04, we report the highest AP score that we are aware of, 0.3697, with our best model BERT(MS MARCO $\rightarrow$ MB). The most effective neural model on Robust04 was reported to be X [] at 0.3124 AP according to the meta-analysis of over 100 papers up until 2019 on this dataset by [][1]. Furthermore, our results with the same model on Robust04 exceed the previous highest

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

known score of 0.3686, which is a non-neural method based on ensembles [13]. This high water mark has stood unchallenged for ten years.

More 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 (3.2), 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.

5.2.3 Number of Sentences

In general, we consider up to three top scoring sentences in each document to compute an overall document score. Our results in Table 5.1 suggest that the top scoring sentence alone is often a good indicator of document relevance; in fact we achieve the best MAP in about half of the experiments by only considering the most relevant sentence. This finding is consistent with those of [67] who verified through user studies that the most relevant sentence or paragraph in a document provides a reliable proxy for document relevance.

Considering the next most relevant sentence in addition to the top sentence yields a noticeable increase across all metrics in some of the experiments. This is especially true in the case of better performing models such as BERT(MS MARCO $\rightarrow$ MB) which might be explained by their ability to effectively match multiple useful sentences. However, we find that adding a third in fact causes effectiveness to drop in some cases. Try for more than 3 for a few and compare Comment for all three metrics and datastes Conclusion?

5.2.4 Per-Query Error Analysis

Our results in Table 5.1 give a good overview of the effect of training data on cross-domain relevance transfer. However, they do not reveal much concerning the particular strengths and weaknesses of each model compared to the baseline. To gain further insight into the

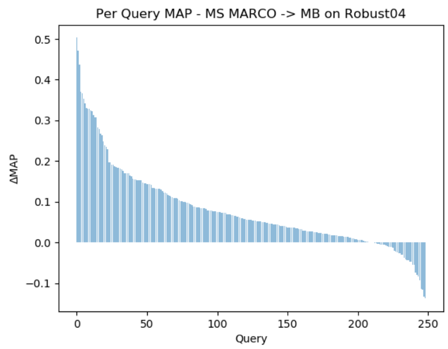

Figure 5.1: first figure

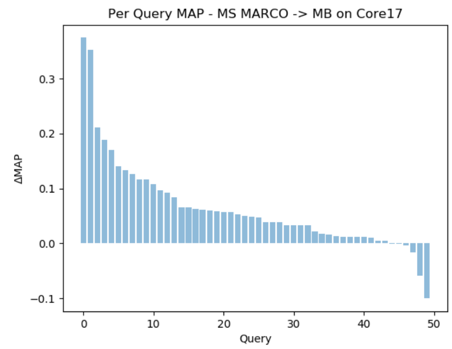

Figure 5.2: second figure

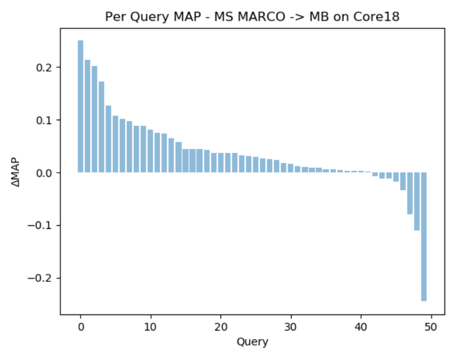

Figure 5.3: second figure

characteristics of our models, we analyze the per-query retrieval effectiveness of each model compared to the baseline on Robust04, Core17 and Core18. Figure X plots Δ AP per query (i.e: $\mathtt{AP}_{\mathtt{Model}} - \mathtt{AP}_{\mathtt{BM25+RM3}}$, sorted in descending order.

TODO

Figure **??**, **??** and **??** show the per query MAP difference between our best model and the baseline. Our best BERT-based model performs better than the baseline for 83% of the queries on Robust04, 88% on Core17 and 84% on Core18. This again confirms that BERT is able to capture relevance signals from MS MARCO and MB which directly help with retrieval on our test collecetions. The best performing queries on Robust04 are `stirling engine`, `human stampede` and `native american casino`, respectively. The worst performing queries are `flavr savr tomato`, `polygamy polyandry polygyny` and `adult immigrants English`.

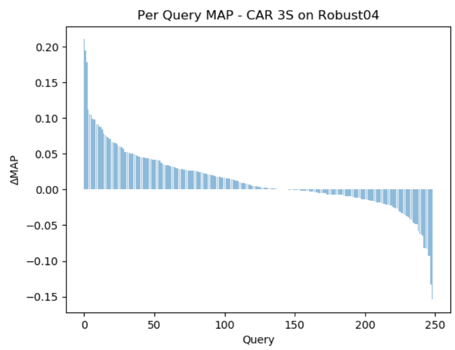

Figure 5.4: first figure

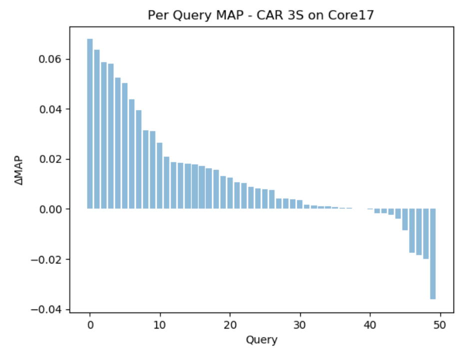

Figure 5.5: second figure

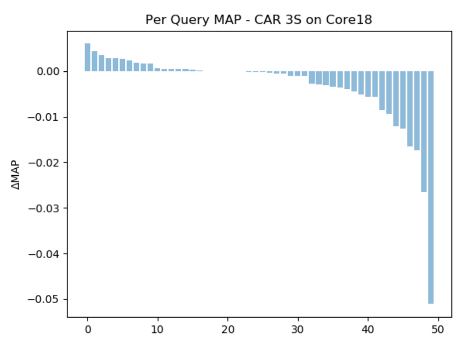

Figure 5.6: second figure

Query	Document ID	
372: native american casino	LA061290-0112	The Sycuan Reservation, one of 18 American India
	FT941-13912	

We repeat the same process for our worst model BERT(CAR) for which we present results in Figures **??**, **??** and **??**. This model outperforms the baseline only for 57% of the Robust04 and 34% of Core18 queries. It performs especially bad for the query 316 `polygamy polyandry polygyny` on Robust04 and 802 `women driving in Saudi Arabia` on Core18.

Overall the query 316 `polygamy polyandry polygyny` appears to be difficult for BERT based models to correctly judge. Different one? In order to compare the characteristics of the two models that cause them to either excel or fail, we look into the ranking of documents for the best and worst performing queries for both. Specifically, we sample sentences from relevant and non-relevant documents for the given query and compare how their rank these documents. Details!

	Query Length				
Model	**1**	**2**	**3**	**4**	**5**
BM25+RM3	0.3889	0.2975	0.2760	0.3089	0.2972
BERT(MB)	0.4145	0.3624	0.3229	0.3681	0.4165
BERT(CAR)	0.3978	0.3112	0.2871	0.3288	0.2844
BERT(MS MARCO)	0.3997	0.3117	0.2872	0.3289	0.2871
BERT(CAR $\rightarrow$ MB)	0.4202	0.3599	0.3296	0.3757	0.4926
BERT(MS MARCO $\rightarrow$ MB)	0.4253	0.3880	0.3480	0.4077	0.5469

Table 5.2: TODO

5.2.5 Effect of Length

Query Length

To investigate the ability of our BERT-based models to exploit context-aware representations of the queries, we study the MAP across increasing query lengths. We conjecture that longer queries would give our models richer context to work with, therefore increasing retrieval effectiveness. To this end, we categorize the 250 queries from the Robust Track by the number of tokens, and evaluate average MAP per each query length, i.e: from 1 to 5. This approach is somewhat limited by the narrow range of query lengths, but still gives insight into the effect of query length on each model's performance.

The results for this experiment are shown in Table 5.2. As expected, the AP for the BM25+RM3 baseline drops by 24% from a single token query to a query with five tokens. Other than that, the models produce improvements over the baseline to similar degrees as when they are evaluated over all queries with BERT(MS MARCO $\rightarrow$ MB) as the best model across all query lengths. Similar to the baseline, BERT(CAR) and BERT(MS MARCO) experience a gradual decline in AP as the query length increases from 1 to 5, leading to an overall 29% and 28% decrease respectively. Note, however, that the decrease in AP is not continuous at each step but reaches a minimum at query length 3 and a maximum at query length 4. It might be helpful to manually inspect rankings for each query length or to use a different averaging technique less sensitive to sample size in order to better reason about this observation; however, we leave this for future work... More interestingly, while increasing query length results in a decrease in AP across all the other models, BERT(MB), BERT(CAR $\rightarrow$ MB) and BERT(MS MARCO $\rightarrow$ MB) in fact perform much better at query length 5 than at 1. Moreover, the performance of the best model BERT(MS MARCO $\rightarrow$ MB) degrades much less where the minimum usually occurs, i.e: query length 3, compared to the baseline, BERT(CAR) and BERT(MS MARCO). What else?

Document Length

It is interesting to note that fine-tuning on MS MARCO or CAR alone results in marginal improvements over the baseline. Considering any number of top scoring sentences, BERT(CAR) and BERT(MS MARCO) both have around 1.2 AP higher than BM25+RM3 on Robust04 with similar gains in P@20 and NDCG@20. Although still statistically significant, these improvements are much lower than those gained with fine-tuning on MB (5 AP).

	Robust04		
BM25+RM3	0.2903	0.3821	0.4407
1S: $\text{BERT}_{\text{Large}}$(CAR')	0.3030†	0.3980†	0.4520
1S: $\text{BERT}_{\text{Large}}$(MS MARCO')	0.3300†	0.4309†	0.4906†

Table 5.3: Ranking effectiveness on chopped MS MARCO and CAR

We attribute this phenomenon to the length mismatch between the training and evaluation text lengths. After dividing into chunks that fit the input span that BERT can handle, the average sentence length in Robust04, Core17 and Core18 sentences is 19 tokens long, which is similar to the average number of tokens in MB training at 15. However, both MS MARCO and CAR give statistics

To validate this hypothesis, we try dividing MS MARCO and CAR passages into chunks the same size as Robust04 sentences. We find that retrieval effectiveness slightly improves upon fine-tuning on the shortened datasets although much less so than expected. From this finding we infer that while comparable document length is an important consideration for cross-domain relevance transfer, there may be other factors at play in terms of X. Future work...

5.2.6 Matching Tendencies

Semantic Matching

To isolate the contribution of BERT, we filter sentences in Robust04 that don't contain any of the query terms. This essentially eliminates the impact of exact matching on the sentence relevance scores, allowing us to verify whether our BERT-based models successfully leverage semantic matching. Table 5.4 displays the retrieval effectiveness of all models on the "pruned" Robust04. Filtering Robust04 sentences leads to a decrease in all metrics across all models, which indicates that exact matching signals are still valuable in relevance predictions. However, notice that the performance of all models still beats the baseline, which implies that they indeed perform semantic matching with notable gains. The improvements over the baseline follow the same trend as the results in Table 5.1. While BERT(CAR) and BERT(MS MARCO) yield minor improvements over the baseline, the best performing model BERT(MS MARCO $\rightarrow$ MB) is significantly higher. Significance testing Interestingly, the drop in AP caused by filtering sentences is also the highest for the best performing model, indicating that this model is able to exploit both exact and semantic matching signals. The overall effectiveness of this model on the original Robust04 dataset shown in Table 5.1 may be owing to the joint matching power demonstrated in this experiment.

Query Terms

BM25 favors documents with frequent occurrences of query terms which is not necessarily the case of our BERT-based models. Hypothesis Therefore, we calculate the fraction of query terms (FQT) in the top k documents for each query across all models, removing stopwords and punctuation. We average the FQT across all retrieved documents for each query:

$$FQT(q) = \frac{1}{k} \sum_{i=1}^{k} \frac{N(d_i, q)}{|d_i|} \tag{5.1}$$

where $N(d_i, q)$ denotes the number of occurrences fo query tokens q in document d_i. limit k to 10 to avoid noise? Divide into 3 buckets instead? FQT average for each model Comment

Because BM25 relies too much on query term frequency, it may rank documents that contain multiple repetitions of query words but do not actually convey any useful information too highly. Therefore, we expect the effectiveness of BM25 to drop more significantly

	Pruned Robust04		
Model	**AP**	**P@20**	**NDCG@20**
BM25+RM3	0.2903	0.3821	0.4407
BERT(MB)	0.3031	0.4014	0.4580
BERT(CAR)	0.2959	0.3936	0.4480
BERT(MS MARCO)	0.2962	0.3936	0.4483
BERT(CAR → MB)	0.3037	0.3998	0.4527
BERT(MS MARCO → MB)	0.3101	0.4102	0.4639

Table 5.4: TODO

	Fraction of Query Term				
Model	**[0.0, 0.01)**	**[0.01, 0.02)**	**[0.02, 0.03)**	**[0.03, 0.04)**	**[0.4, 0.05)**
BM25+RM3	0.2973	0.2863	0.2890	0.1961	0.1152
BERT(MB)	0.3436	0.3628	0.3548	0.2793	0.1314
BERT(CAR)	0.3082	0.3094	0.3121	0.1902	0.1076
BERT(MS MARCO)	0.3088	0.3095	0.3112	0.1883	0.1087
BERT(CAR $\rightarrow$ MB)	0.X	0.X	0.X	0.X	0.X
BERT(MS MARCO $\rightarrow$ MB)	0.3880	0.3809	0.3028	0.X	0.X

Table 5.5: TODO

with increasing FQT. To observe how retrieval effectiveness changes across different ranges of FQT, we split the values into X buckets and calculate the average MAP for each bucket. Comment Give examples of useless high FQT

5.2.7 Novel Terms

Pretrained on massive amounts of data for language modeling, BERT can capture various semantic relationships helpful for relevance prediction. Figures X one of the top scoring sentences for the query 322 `international art crime`. Note that the sentence does not contain any exact matches with the query terms and is nonetheless relevant to the query.

description This highlights the power of...

MS MARCO MB

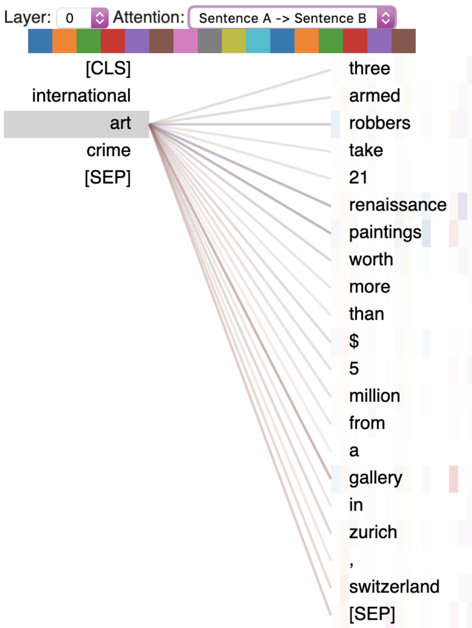

Figure 5.7: first figure

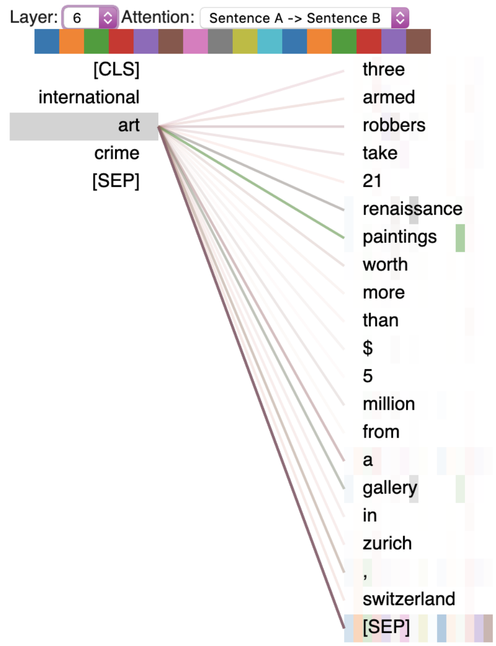

Figure 5.8: second figure

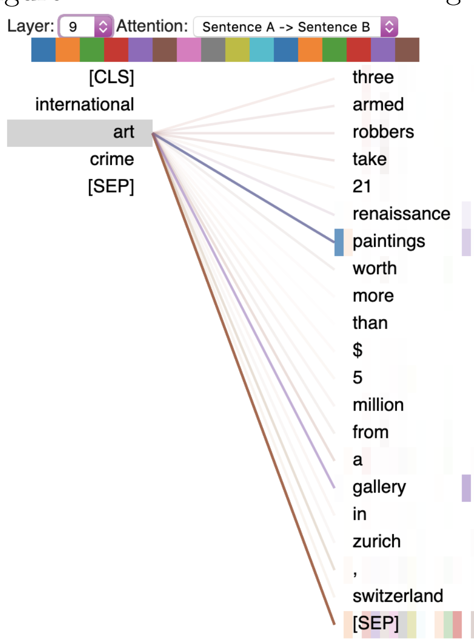

Figure 5.9: second figure

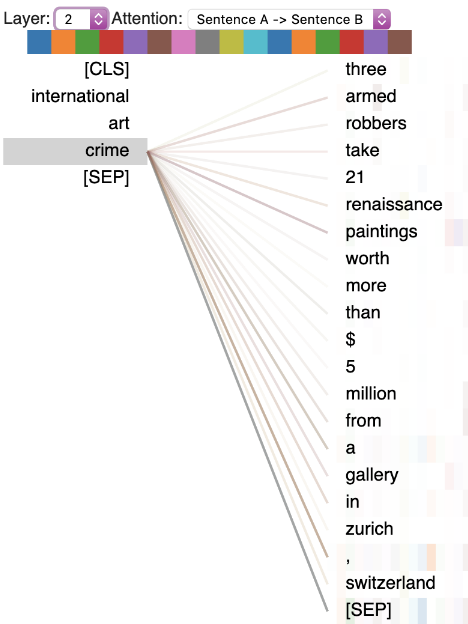

Figure 5.10: first figure

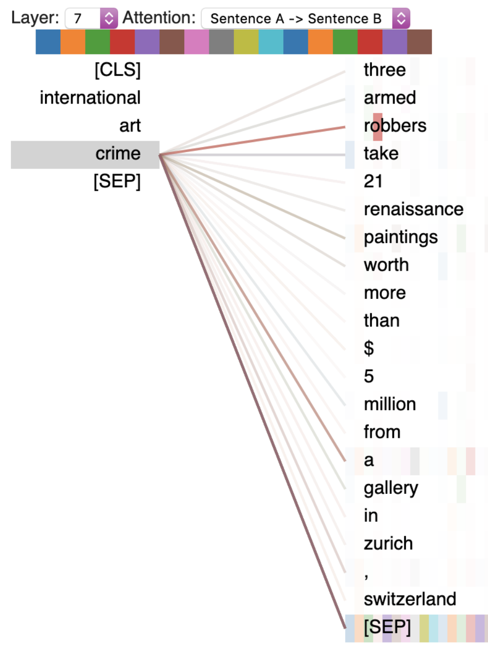

Figure 5.11: second figure

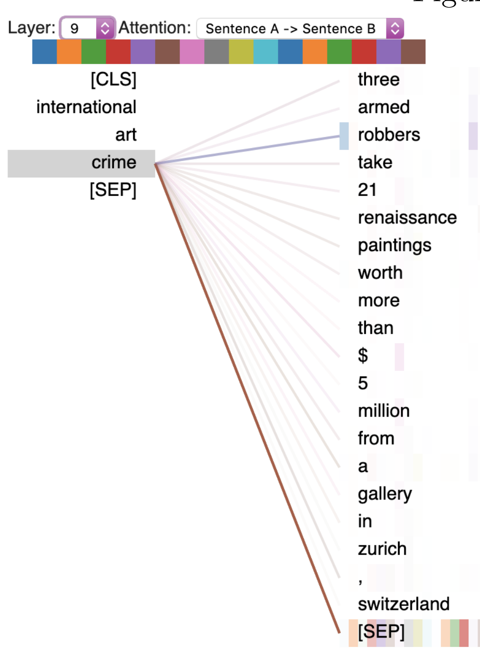

Figure 5.12: second figure

Chapter 6

Conclusion and Future Work

In this thesis, we propose two innovations to successfully apply BERT to document retrieval with significant improvements on three TREC newswire collections. To overcome the maximum input length restriction imposed by BERT, we focus on integrating sentence-level evidence to rank newswire documents. This approach requires sentence-level relevance labels to train BERT as a relevance classifier; however, relevance judgements in most test collections are provided only at the document level. We address this challenge by leveraging sentence- and passage-level relevance judgements fortuitously available in out-of-domain collections. More specifically, we fine-tune BERT with the goal of capturing cross-domain notions of relevance.

We show that relevance models learned with BERT can indeed be transferred across domains in a straightforward manner. Combined with sentence-level relevance modeling, our simple model achieves state-of-the-art results across all three newswire collections. Furthermore, our results suggest that document ranking can be essentially distilled into relevance prediction at the sentence level. X

A promising future direction based on our findings includes ... X

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] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview.

[11] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[12] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*, 2019.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

[23] 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.

[24] Ganesh Jawahar, Benoît Sagot, Djamé Seddah, et al. What does bert learn about the structure of language?

[25] 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.

[26] 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.

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

[28] Ping Li, Qiang Wu, and Christopher J Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2008.

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

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

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

[32] Tie-Yan Liu. *Learning to rank for information retrieval.* Springer Science & Business Media, 2011.

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

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

[35] 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.

[36] 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.

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

[38] 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.

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

[40] Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. Overview of the trec-2011 microblog track.

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

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

[43] 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.

[44] 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.

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

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

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

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

[49] 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.

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

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

[56] 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.

[57] 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.

[58] 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.

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

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

[61] 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.

[62] 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.

[63] 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.

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

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

[66] 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.

[67] 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.

[68] 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.

[69] 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.