Chapter 1

Cross-Domain Relevance Transfer with BERT

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

1.1 Datasets

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 1.1: TODO

1.1.1 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. [**?**]. 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 1.1: TODO

where ("irrelevant", "relevant" and "highly relevant"); however, for the purposes of this work we treat both higher degrees of relevance as equal [**?**]. 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.

1.1.2 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 [**?**] 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 1.2: 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 1.2: 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. [**?**], 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

1.1.3 TREC Complex Answer Retrieval (CAR)

TREC CAR [**?**] 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?

1.2 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 [**?**, **?**]. More on this? This phenomenon allows a number of NLP tasks to greatly benefit from featured implicitly encoded in BERT weights.

1.2.1 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

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 1.3: 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 1.3: TODO

blahblahblah

Figure 1.4: Put tokenized BERT input example?

or passages from MS MARCO or TREC CAR. Following Nogueira et al. [**?**], 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 $[\texttt{[CLS]}, q, \texttt{[SEP]}, d, \texttt{[SEP]}]$. 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

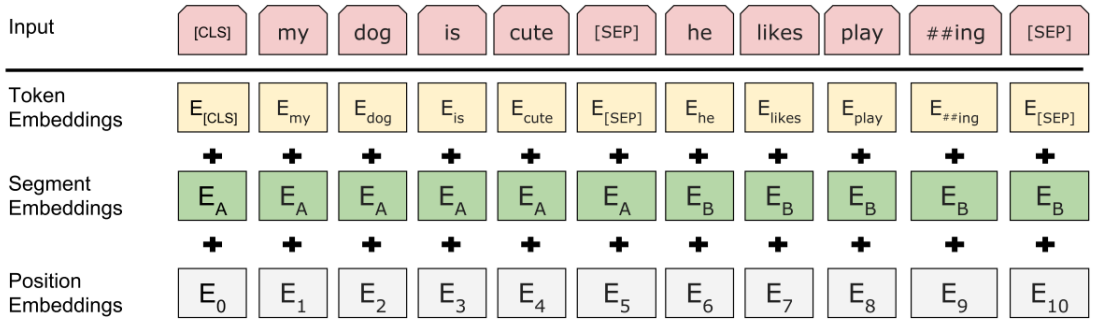

Figure 1.5: Create customized diagram

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:

$$\sigma(y_j) = \frac{e^{y_j}}{\sum_i e^{y_i}} \tag{1.1}$$

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.

1.3 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 1.6: ...

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:

$$S_f = a \cdot S_{doc} + (1 - \alpha) \cdot \sum_{i=1}^{n} w_i \cdot S_i \tag{1.2}$$

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?

1.4 Experimental Setup

1.4.1 Training and Inference with BERT

We fine-tune $\text{BERT}_{\text{Large}}$ [?] 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 [**?**]. We use Adam [**?**] 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 [**?**] 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.

1.4.2 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 [**?**] 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.