RADBOUD UNIVERSITY

MSc THESIS

Have a chat with BERT; passage re-ranking using conversational context

Author:
T. CRIJNS

Supervisor:
Prof. dr. ir. A.P. DE VRIES

Second Reader:
Dr. ir. F. HASIBI

A thesis submitted in fulfillment of the requirements for the degree of Master of science in Computing Science

Institute of Computing and Information Sciences
Faculty of Science

August 22, 2019

Abstract

In this thesis, the problem of question answering using conversational context is explored for the Conversational Assistance Track of the Text REtrieval Conference. The goal of the task in this track is to provide an answer to a query by retrieving responses from a large set of short text passages and ranking these according to their computed relevance to the query. The state-of-the-art NLP model BERT [1] is compared to the baseline ranker BM25 [2]. It is researched how including conversational context in the input influences the predictive power of these models. Multiple methods for dealing with the input restrictions of the BERT model are compared; clipping, summarization and, rank and score fusion. This was evaluated on two datasets of a different nature; a dataset containing artificially generated conversational sessions and one containing handcrafted, more human-like sessions. The results show that BERT outperforms BM25 in all experiments, regardless if or how much context is taken into account. Adding conversational context to the input however does not increase predictive power in both models. Nonetheless, the context processing methods can still be compared; the fusion methods perform best. Additionally, it does not matter for the experiments in this works whether the data is constructed artificially or in a more realistic manner.

Contents

Chapter 1

Introduction

1.1 Conversational assistance

Conversational assistance is a relatively new research area that is expanding together with the increasing use of smart speakers and speech assistants such as Google Assistant, Amazon Echo and Apple HomePod. These assistants are equipped with technologies that are able to simulate human conversation. In technologically advanced societies, it is common to encounter such artificial conversation partners, as the underlying technologies have existed in various forms for some time since as early as 1966, when the pioneering chatbot ELIZA was created in the MIT AI Laboratory [3]. ELIZA was one of the first programs thought to have a chance of passing the Turing Test [4]. Users could communicate with ELIZA through written text, and scripts and predetermined rules were used to formulate responses.

Modern-day assistants surpass this basic form of communication. They are equipped with state-of-the art hardware and natural language processing techniques in order to provide the user with a realistic conversational experience. A survey on the usage of smart speakers has shown that many people interact with these systems; 26% of internet users in the US and 22,4% in the UK own a smart speaker[1]. However, online reviews and public opinion reports show that the performance of such systems can still be considerably improved[2]. For well-defined actions, the assistants seem to perform well. When tasks become more complex, their performance declines. In order for these assistants to outgrow their current 'nice to have' status and become must haves, they have to become less awkward to interact with and demonstrate advantages over established information retrieval techniques.

In the digital age, searching for information is mainly conducted in a non-verbal and non-conversational manner. Google is currently the most used search engine worldwide[3] and it is tasked with processing 40.000 queries per second[4]. It has become so popular that the verb 'Google' was added to the Oxford English Dictionary in 2006[5]. Googling has become the state of the art in online information finding. In its most basic form, googling can be described as follows: a user tries to satisfy their need for information by submitting a single query to the search engine. The user is then presented with a set of search results from which they can choose. Satisfying a search query with a single answer instead of a set, which is the goal in conversational search, is considerably more difficult. If the user is no longe able to select the best result, they must be provided with highly relevant search results.

[1] www.emarketer.com/content/global-smart-speaker-users-2019 (Last accessed: 2/5/2019)
[2] www.marketwatch.com/story/heres-how-smart-speakers-need-to-get-smarter-2018-09-20 (Last accessed: 4/5/2019)
[3] http://gs.statcounter.com/search-engine-market-share (Last accessed: 16/6/2019)
[4] www.internetlivestats.com/google-search-statistics (Last accessed: 16/6/2019)
[5] www.fool.com/investing/dividends-income/2006/07/05/to-google-or-not-to-google.aspx (Last accessed: 16/6/2019)

The task of providing a single relevant search result is only one in a large set of important challenges associated with building a realistic and sophisticated conversational assistant. The Alexa Prize[6], an academic competition launched by Amazon to further research in conversational assistance, indicates that other challenges include; *natural language understanding*, *context modeling*, *dialog management*, *response generation* and *knowledge acquisition*.

1.2 Task definition

This study was conducted in the context of the Conversational Assistance Track (CAsT)[7] of the Text REtrieval Conference (TREC)[8]. This is the opening year of the track, which was established to promote research on conversational search. This type of search is a form of human/computer interaction where a user can ask a device a question in verbal or written form, and the device is able to provide a relevant response in the form of a sentence. The main goal of this interaction is to satisfy the user's informational need. As is the case with human conversation, a discourse with a conversational assistant is often expressed through multiple conversation turns.
The goal of the task is to produce a proper response to a query, given the previous conversation turns as context. In this case, the production of a proper response involves retrieving responses from a large set of short text passages (1-3 sentences in length) and ranking these according to their computed relevance to the query. In summary: the task is text passage retrieval and re-ranking for question answering, given conversational context. This research focuses on only a subsection of the track, which is further explained in Section 1.4.

1.3 Background and related work

In this section, the theoretical framework on which this research is based is introduced. The research area in focus is question answering. The task at hand extends this domain with the addition of multiple conversation turns as context for a question.

1.3.1 Early work in question answering

At around the same time that ELIZA was built, other question answering programs, such as Baseball [5], emerged as well. These systems mainly focused on question answering, whereas ELIZA was built to provide a convincing simulation of a human conversation. Baseball was able to answer questions about the past year's baseball season in the United States. These early systems worked fairly well, primarily due to the existence of a source database handcrafted by experts on the systems' respective topics. Often, a rule-based system was used to retrieve information from these databases, and the information was formatted using scripts.

[6] www.developer.amazon.com/alexaprize (Last accessed: 4/5/2019)
[7] www.treccast.ai (Last accessed: 5/8/2019)
[8] trec.nist.gov (Last accessed: 13/7/2019)

In the 1990s, question answering was identified as a problem that could be approached with information retrieval techniques. In 1999, the first question answering track was established at TREC [6]. At that time, most participants used a multistep strategy to approach the question answering problem, and each step constituted its own separate research domain. The following three steps were most common:

1. Question classification: identify what type of question the input is, for example a 'who' question or a 'what' question.
2. Candidate retrieval: use basic text retrieval algorithms to find documents that might be relevant to the input.
3. Answer retrieval: filter answers that do not match the question type identified in step 1 and select the most relevant answers from the candidates found in step 2.

A similar version of this general multistep approach is still used today. An example is the architecture that Ji et al. [7] proposed to deal with the short text conversation problem. The task in this problem is for a machine to return a reasonable response to a short message provided by a human. Their setup also consists of multiple steps: first, they use basic linear matching models based on, for example, *term frequency-inverse document frequency* (TF-IDF) similarity, to retrieve a candidate document set. In the second step, they add more features to each candidate and use a linear ranking function to evaluate the entire set.
A lot can be gained in this second part: while it is relatively simple to find candidates that are in some way connected to the initial query, it is difficult to determine which are most relevant. In order to do so, the relation between the possible output candidates and the input queries must be understood on a deep linguistic level. The focus of research in this area has shifted from the above mentioned rule-based approaches to machine learning methods.

1.3.2 Machine learning in NLP

The focus of this study is on the task of evaluating and ranking candidate passages using machine learning methods. The relevant models are introduced after first discussing their precursors.

Recurrent neural networks

Textual data has the property that the shape and order of all data points carry meaning, such data is best processed sequentially. *Recurrent neural networks* (RNNs) [8] were introduced in 1990, and their structure aligns perfectly with the sequential nature of textual data. RRNs are called *recurrent* because they execute the same task for every item in a sequence, with the output of each task being dependent on the output of previous calculations. This type of network has the ability to hold information of these previous calculations in memory whilst processing new information. For a long time, RNNs were the most popular architecture for a wide variety of language processing tasks.
One specific type of network, called a *long short-term memory network* (LSTM) [9], outperforms the original RNN framework on many such tasks. This network is able to 'remember' information for longer periods and learn long-term dependencies. Tan et al. [10] have addressed the problem of answer selection using bi-directional LSTMs. They built embeddings of all questions and answers using these models and compared them based on the cosine similarity of the word vectors.

Attention mechanisms

Recurrent neural networks were later enriched with mechanisms such as neural attention. The main aim of adding attention to these models is to restrict the focus of the computation to the information that is needed for the task at hand. For example, when processing a query and answer, the attention mechanism has access to all the information about previously processed input. The mechanism calculates the attention for each word in relation to other words in the input, denoting the relevancy to that word.

A visual example of attention in a model processing textual data is depicted in Figure 1.1. This figure shows the attention weight matrix in a sentence translation task; the lighter the cell, the higher the attention activation[9].

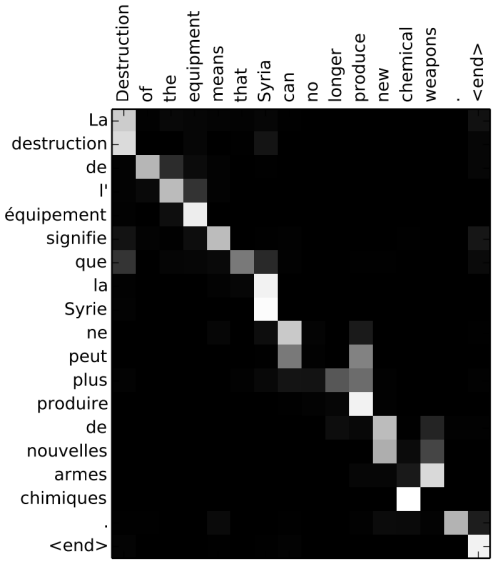

FIGURE 1.1: Neural attention weight matrix example in a sentence translation task

Shen et al. [11] have proposed the Knowledge-Aware Attentive Bidirectional Long Short-Term Memory unit (KABLSTM) that adopts external information from knowledge graphs to enrich the representations of question and answer pairs. Furthermore, a knowledge-aware attention mechanism is presented to attend to interrelations between each segment of question and answer pair. Wang et al. [12] put forward an *RNN* that computes attention before calculating the hidden sentence representation, whereas previous attention based RNNs added attention after. They demonstrate that this form of adding attention which they call *inner attention* outperforms the older strategy, called *outer attention*. Wu et al. [13] have proposed a *topic-aware attentive RNN* in which representations of a message and the response are enhanced by the information on the topic at hand. The attention mechanism uses the topic information to refine the representations of the message and the response. Yang et al. [14] have proposed an attention-based neural matching model for ranking short answer text. This model learns weighted representations of question and answer pairs using attention based similarity matrices.

In 2017, a new technique introduced a new standard to the state of the art in neural NLP. In their paper 'Attention is all you need', Vaswani et al. [15] proposed a network architecture, the *Transformer*, that is solely based on attention mechanisms. Recent studies have attempted to improve upon this attention-based network. For example, Zhou et al. [16] proposed extending the attention mechanism in two ways: by constructing representations of text using only stacked self-attention, and by trying to extract the truly matched segment pairs with attention across the context and response. Self-attention is a specific form of attention where a sequence attends to

[9] http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/ (Last accessed: 4/8/2019)

itself, the goal is to learn the internal structure and the dependencies between the elements. Using textual data, this could give a deeper understanding of the grammatical structure of a sentence and the intra word-level dependencies. They integrated the two methods into one neural network to achieve state-of-the-art results.

BERT

One year after the introduction of the Transformer architecture, another significant breakthrough in performance was made by the natural language representation model BERT, which stands for Bidirectional Encoder Representations from Transformers. Devlin et al. [1] created this model, which was designed to pre-train word representations in a bidirectional manner, and is based on the Transformer architecture. With these pre-trained representations, state-of-the-art results were achieved in a wide range of tasks by fine-tuning the model using only one additional layer. BERT builds on clever solutions regarding transfer learning in NLP, such as ELMO [17] and the openAI Transformer [18]. Noguira et al. [19] used BERT as a passage re-ranker and achieved state-of-the-art-results for the MS MARCO passage re-ranking task[10] and on the TREC-CAR[11] dataset from 2017.

1.3.3 Utilizing conversational context in question answering

A conversation can be seen as an extended question answering session. In order to formulate a proper response to the last query of a session, the preceding utterances can be taken into account. Lowe et al. [20] simply concatenate the context utterances and add them to the input. This adds a large amount of additional information, but also a lot of noise. Not all conversational context is relevant for retrieving the right answer, and sometimes the context is not needed at all. Two recent works have addressed this issue by attempting to select the right context instead of using all of it. Zhang et al. [21] have proposed a deep utterance aggregation approach to obtain a representation of the relevant context. This approach weighs the previous utterances in the context to reduce noise. Yan et al. [22] have proposed a contextual query reformulation framework that utilizes rank fusion.

1.4 Research approach

In the question answering domain, BERT is currently topping the leaderboard in natural language challenges such as the MS MARCO passage re-ranking task[10] and the question answering challenge featuring multi-hop questions[12]. This thesis investigates whether BERT can also be utilized for a re-ranking task using conversational context.

In a recent study by Padigela et al. [23], the successes and failures of using BERT as a passage re-ranker in regular question answering are investigated. The authors compared BERT's performance to that of BM25. They showed that BERT performs well on the MS MARCO passage re-ranking task but conclude that despite this good performance, the encoding of the query context for longer queries can be improved. Having a good encoding of the query context is even more relevant in passage re-ranking for conversational data because of the size of the input, which can be much

[10] www.msmarco.org/leaders.aspx (Last accessed: 5/8/2019)
[11] trec-car.cs.unh.edu (Last accessed: 27/7/2019)
[12] hotpotqa.github.io/ (Last accessed: 23/7/2019)

larger than in regular question answering. In addition, due to input restrictions in deep learning toolkits, the BERT model is also physically restricted to an input size of 512 tokens. Thus, it is necessary to find a way to fit in all of the context so that it meets the BERT input restrictions in the best way possible.

To investigate the workings of BERT in a conversational passage re-ranking task, a study similar to that of Padigela et al. was conducted for this thesis. In this study, BM25 is also used for baseline comparison, as it is an efficient, intuitive and simplistic ranking function. The predictive power of both BM25 and BERT and their ability to re-rank candidate passages to obtain the best answer passage to a query is examined. The study's main research question is formulated as follows:

> **RQ1** - How does BERT perform in a conversational re-ranking task in comparison with the baseline ranker BM25?

The following sub-questions are also identified:

> **RQ1.1** - To what extent does predictive power change when adding conversational context to the input?
>
> **RQ1.2** - How can BERT's input restrictions (512 tokens) be dealt with when using conversational data?

The first dataset consists of artificially generated query sessions, and the second contains more realistic query sessions based on human conversation (see Chapter 2 for further detail). It is hypothesized that artificially generated data lacks crucial connections between the utterances that are probably present in more realistic conversational data. Therefore, an additional sub-question is addressed to further validate our results:

> **RQ2** - Does the value of context differ across an artificially generated conversational dataset and a more realistic conversational dataset?

1.5 Structure of the report

This report features six chapters. As could be read above, Chapter 1 introduces the problem domain of conversational assistance, the relevant literature and background information associated with methods in this field and the research questions. Both Chapter 2 and 3 give the reader an overview of all technical background information that is needed to understand the experiments conducted in this research. Chapter 2 describes the two datasets that are used, and provides examples of sessions from both datasets. Chapter 3 explains the inner workings of all techniques used in the experiments. Chapter 4 gives an in depth description of how the experiments were established, and also provides technical implementation details. Chapter 5 presents the results that were achieved in the experiments, and an interpretation of these results. Chapter 6 describes the results in in light of the research problem, the shortcomings of this work, and gives suggestions for future work. Chapter 7 concludes this thesis by reflecting on the research process, methods and results. Lastly, the research questions are answered.

Chapter 2

Data

This chapter describes the data used for the experiments. As part of the track, which was explained in Section 1.2, conversational data was provided to use for training and development. There are very few conversational datasets available, and so it is also a goal of the track to create a reusable benchmark dataset for research in the conversational domain. The data provided by the track consists of multiple data sources. This study used the MS MARCO conversational session data, as it contains a large number of artificial query sessions that also have relevance labels. The second dataset that was used is a small set of realistic query sessions based on human conversation, hand-labeled by the track organizers. The limitations of both datasets are discussed in Chapter 6.

2.1 MS MARCO - Bing Informational Sessions

On the 23rd of April 2019, Microsoft released a conversational search dataset[1]. The dataset consists of 45,040,730 artificially created query sessions. These sessions were generated using embedding techniques and a clustering algorithm to collect and reassemble the 1,010,916 already existing MS MARCO queries. An example of a synthetic query session is provided in Figure 2.1:

Figure 2.1. Example of a generated query session

Title: marco-gen-dev-40
Queries:

- What is the australian flag?
- What is the population of australia?
- What hemisphere is north australia?
- How big is sydney australia?
- Is australia a country?

This dataset does not contain any relevance labels; it only contains query sessions. However, because the sessions are a cluster of single queries that have a relevance label, the labels can be obtained by utilizing the regular MS MARCO passage reranking dataset. The sessions dataset was merged with this dataset to obtain query identifiers and the relevance labels. Sessions with fewer than five queries were discarded to ensure that there was sufficient context to work with. One thousand of these synthetic sessions were randomly selected to use as the test set.

[1] http://www.msmarco.org/dataset.aspx (Last accessed: 5/8/2019)

2.2 TREC CAsT sessions

The organizers of CAsT released a small dataset, which contains 80 conversational sessions. The track organizers hand-labeled the queries of 13 sessions with answer passages from three passage sets: the MS MARCO collection, the Washington Post Corpus[2] and the TREC CAR paragraph collection[3]. Each query in the sessions has labels for 20 answer passages, which can be either 2, 1 or 0. These labels denote *very relevant*, *relevant* and *not relevant*. To be able to answer **RQ2**, this dataset was used as a comparison to the MS MARCO dataset. An example of a session is provided in Figure 2.2:

Figure 2.2 Example of a manually created dialogue

Title: US Judicial history
Description: Judicial history in the US including key court cases and what they established.
Queries:

- What are the most important US Supreme Court cases?
- Was it unanimous?
- What was the implication of roe vs wade?
- What were the main arguments?
- What was the point of the brown v board of education?

[2]https://trec.nist.gov/data/wapost/ (Last accessed: 23/7/2019)
[3]trec-car.cs.unh.edu/datareleases/ (Last accessed: 23/7/2019)

Chapter 3

Methods

This chapter explains the inner workings of BM25, BERT, extractive summarization, rank fusion and score fusion. These are all techniques used in the experiments in this work.

3.1 BM25

BM25, or *Okapi BM25* [2], is a ranking function that is used to match documents according to their relevance to a query. BM stands for 'Best Matching'. The foundation of BM25 is the TD-IDF weighting. TF-IDF [24] is a technique that weighs a term's frequency (TF) in a document and its inverse document frequency (IDF). This weighting identifies words that are most unique to a document. For example, when using an English document corpus, the word 'the' will occur with a high frequency in all documents, and so its TF-IDF score will be low. The formula for this weighting is given in Equation 3.1:

$$w(t,d) = tf \cdot log\frac{N}{df} \tag{3.1}$$

In this formula, $w(t,d)$ is the weight of document d for term t, N is the total number of documents and df is the number of documents that contain the term t (the document frequency). The TF-IDF weighting scheme can be extended to BM25 by adding an additional weighting for document length. The formula for BM25 is given in Equation 3.2:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})} \tag{3.2}$$

This formula can be used to calculate a BM25 score between query Q and document D. We sum over all the keywords $q_1, ..., q_n$ in the query. $IDF(q_i)$ is the inverse document frequency weight of a term q_i. $f(q_i, D)$ is the term frequency of a term q_i in the document D. $|D|$ is the total number of words in the document D. $avgdl$ is the average document length of all documents. k_1 and b are freely chosen parameters.

3.2 BERT

The first part of this section focuses on the underlying architecture on which BERT is based. Subsequently, the innovative techniques introduced by Devlin et al. [1]. are discussed.

3.2.1 The Transformer architecture

BERT uses the Transformer architecture [15] as its underlying structure. The Transformer is an architecture that relies solely on the attention mechanism and removes the need for recurrence by processing the entire input string at once instead of in a word-by-word fashion.

The architecture of the Transformer is shown in Figure 3.1. This representation is simplified, as only a basic understanding of the Transformer is necessary to understand its role in BERT's architecture.

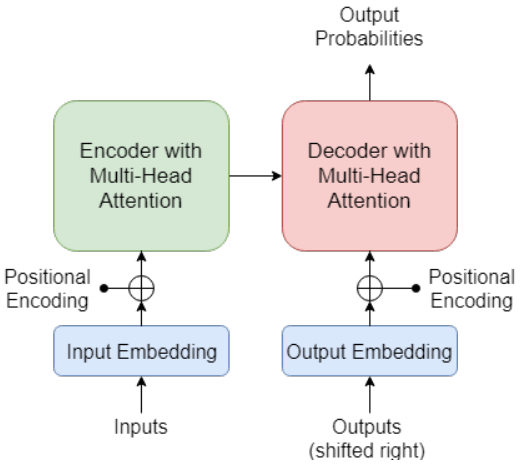

FIGURE 3.1: Simplified Transformer architecture, based on a figure from the original paper [15]

As can be seen, the data traverses the network as follows: the input and output tokens are converted to vectors using learned embeddings. Because all tokens are processed simultaneously, positional information is lost. This issue is solved by adding vectors that carry information regarding the relative or absolute position of the tokens. These vectors are called *positional encodings*.

The model consists of an encoder-decoder structure. Both the encoder and the decoder consist of N stacked *Feed-Forward and Multi-Head Attention layers*.
The inputs to the encoder first proceed through the Multi-Head Attention layer, which consists of self-attention layers that examine the other words in the input sentence while encoding one specific word. The encoded input then moves to the Feed-Forward layer.
The decoder consists of the same elements. However, between the Feed-Forward and the Multi-Head Attention layer, an additional attention layer has been inserted to focus on the most relevant parts of the input sentence.

The Transformer architecture is designed to perform machine translation experiments. However, the authors have shown that it generalizes to other natural language processing tasks as well. The activations in the output can be used for tasks such as speech recognition, summarization, question answering, language understanding, and many more.

3.2.2 The BERT architecture

BERT utilizes only the encoder from the Transformer architecture, as the goal of the network is to generate representations. These representations are trained bidirectionally, which means that for the encoding of each word, the words to the right and left of it in the input sequence are evaluated. The training of a BERT model is similar to the traditional training of word embedding models. In this traditional setting, training often has a specific goal that has to be determined beforehand. For example, many models try to predict the next word in a certain sequence, which limits the usage of these models to that specific task. The authors of BERT have devised sub-tasks that help make BERT a more generally applicable model, these tasks are described below.

Task 1: *Masked language model (Masked LM)*
During training, 15% of the tokens are selected at random and replaced by a [MASK] token. The model then attempts to predict the original token, taking the other tokens in the input into consideration.

Task 2: *Next sentence prediction*
During training, pairs of sentences are fed to the model as input, and the task is to predict whether the second sentence is a logical successor to the first sentence. Half of the training samples are subsequent pairs, and the other half are randomly joined sentences from the corpus. An example of an input for the Next Sentence Prediction task is presented in Figure 3.2.

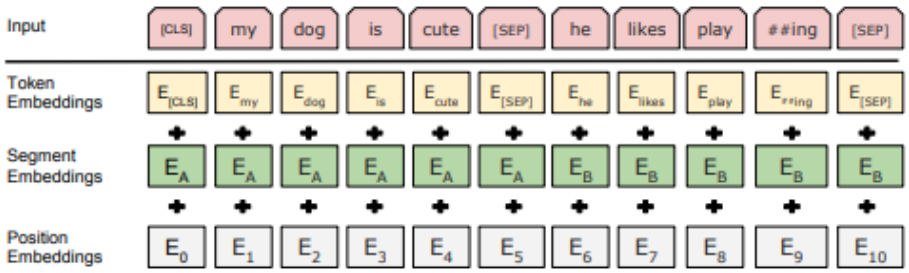

FIGURE 3.2: BERT input representation [1]

A [CLS] token is added to the beginning of the first sentence and a [SEP] token is added to the end of each sentence. Furthermore, a segment embedding vector is added to indicate which tokens belong to first sentence and which to the second sentence. Finally, a positional embedding vector is added to capture the positional information of the sequence words.
With these tasks, BERT becomes less prone to overfitting and is better equipped to handle errors and previously unseen word combinations.

The authors report the performance of two differently sized BERT models, BERT_{BASE} and BERT_{LARGE}. The size refers to the number of Transformer blocks that the model uses. BERT_{BASE} has 12 blocks and a total number of 110 million parameters. BERT_{LARGE} has 24 blocks and a total number of 340 million parameters. The authors pre-trained these models on the tasks mentioned above using the BooksCorpus [25] (800 million words) and English Wikipedia (2500 million words).

3.2.3 Fine-tuning

After explaining the model and the pre-training, the authors demonstrate that, through fine-tuning, BERT can be applied to various different tasks in the natural language processing domain. The fine-tuning involves adding an additional layer on top of the pre-trained BERT model. The authors show an improvement of state-of-the-art-performance in seven of the eleven NLP tasks that they describe.

3.3 Context processing

BERT can take a maximum of 512 input tokens. If information from the context of a query needs to be taken into account, the input size will increase together with the number of conversation turns. In order to capture all the important information from the input conversation, five different methods for fitting inputs that exceed the maximum amount of tokens are considered here. The subsections hereafter elaborate on these methods.

3.3.1 Clipping

Clipping was used as the first method and baseline. In this method, the context was simply concatenated with the query to be answered and if the result was larger than the input size, the context was clipped to fit the 512 tokens.

3.3.2 Extraction-based summarization

Extraction-based summarization is a technique in which important words from a document are extracted and combined to produce a summary. The extraction can be based on any metric, but for this study, the TF-IDF score, explained in section 3.1, was used. Similarly to the clipping method, the context was concatenated with its respective query. When this resulted in an input size that was too large, a number of important words were extracted from the context based on their TF-IDF scores so that the context would fit within the input restrictions.

3.3.3 Rank and score fusion

Rank and *score fusion* methods are techniques in which multiple sets of scores or rankings are combined into one. If the computation is based on rankings alone, it is called *rank fusion*. If the computation involves the fusion of scores on which a ranking can be based, it is called *score fusion*. Instead of transforming the context to fit the input restrictions and performing one run of the model, multiple runs were performed with the query and each of the conversation turns as input. The fusion can be done in many different ways. Three different methods are discussed below.

The first two methods to be discussed are basic score fusion operations, namely average score fusion and maximum score fusion. In this process, multiple sets of scores are obtained and combined by taking either the average of the maximum of the scores. After the scores have been fused, a ranking can be determined. Table 3.1 contains an example with the calculations for three documents (D1, D2 and D3) and three different sets of scores (S1, S2, S3).

	S1	S2	S3	AVG	AVG rank	MAX	MAX rank
D1	0.3	0.5	0.9	0.567	1	0.9	1
D2	0.4	0.6	0.5	0.5	2	0.6	3
D3	0.7	0.2	0.1	0.33	3	0.7	2

TABLE 3.1: Examples of average score fusion (AVG) and maximum score fusion (MAX)

The third method, which was introduced by Cormack et al. [26], is called Reciprocal Rank Fusion (RRF). This is a simple method that sorts the documents according to the formula given in Equation 3.3:

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)} \tag{3.3}$$

D is a set of documents and R a set of rankings. k is a freely chosen parameter for which the authors propose to use 60, they based this number on the results of pilot experiments that indicated that $k = 60$ was near-optimal. Table 3.2 presents an example with the calculations for three documents (D1, D2 and D3) and three different sets of rankings (R1, R2, R3).

	R1	R2	R3	RRF R1	RRF R2	RRF R3	RRF	RRF rank
D1	3	1	2	$\frac{1}{60+3} = 0.159$	$\frac{1}{60+1} = 0.164$	$\frac{1}{60+2} = 0.162$	0.485	2
D2	2	3	3	$\frac{1}{60+2} = 0.162$	$\frac{1}{60+3} = 0.159$	$\frac{1}{60+3} = 0.159$	0.48	3
D3	1	2	1	$\frac{1}{60+1} = 0.164$	$\frac{1}{60+2} = 0.162$	$\frac{1}{60+1} = 0.164$	0.49	1

TABLE 3.2: Examples of reciprocal rank fusion (RRF)

Chapter 4

Experimental setup

This chapter outlines the experimental setup for the comparison of BERT and BM25. The main goal is to compare the techniques when they are applied to solve the problem of passage re-ranking, including experiments using the conversational context. First, the experiments carried out for both of the datasets are described. Thereafter, the implementation details and evaluation metrics are specified.

4.1 Experiments on the MS MARCO dataset

Calculating the BERT activations for a query and answer pair is a costly and time-intensive process. In an ideal situation without restrictions, BERT could be used to evaluate all the possible passages for each input query and its context. This was however not feasible given the available time and means. Therefore, a two-step process similar to the standard approach mentioned in Section 1.3.1 was employed.

For each session in the test set, one fixed query was selected as the query to be answered. Based on this fixed query, candidate passages were retrieved using BM25. The amount of candidate passages to be retrieved per query was chosen by evaluating the number of passages retrieved with a positive label and the number of passages retrieved in total. It is a trade-off between test size and computation time required for running the evaluation on the test data. The best trade-off was found in retrieving 200 candidate passages for each query, and for that amount, BM25 was able to retrieve a passage with a positive label for 582 of the 1,000 initial sessions. Because it does not make sense to evaluate the re-ranking of passages that are irrelevant, the other 418 sessions were discarded.
These relevant passages were then re-ranked using both BM25 and BERT. In order to determine the extent to which the conversational context influenced the models' performance, this approach was repeated with inputs ranging from one conversation turn to ten conversation turns. The test set consists of sessions that contain at least five queries.

As more conversation turns were added to the input, the limitations of BERT's input size had to be dealt with. This is not an issue for BM25, which is bag-of-words based. The six experiments for re-ranking the 200 candidate passages are listed hereafter, and the experiments that require context processing are illustrated by the processed result of the example data presented in Figure 4.1.

- **BM25**: Ten runs with the respective number of conversation turns in the input. The input for BM25 consists of n conversation turns concatenated with the query.

Figure 4.1. Example data for illustration of the processing methods

Context 1: Who is Elmo? Elmo is a red muppet from Sesame Street.
Context 2: Who is BERT? BERT is an NLP model.
Query: Does he know Elmo?
Candidate passage: BERT does not know Elmo.

Maximum number of input tokens allowed: 25

- **BERT + clipping**: Ten runs with the respective number of conversation turns in the input. The input for BERT + clipping consists of n conversation turns concatenated with the query and a candidate passage. Whenever this input failed to meet the input restrictions, the context was clipped so that it fit. The processed example input is depicted in Figure 4.2. Note that in the example, the tokens are equal to the words. However in reality, the BERT tokenizer can create a different representation.

Figure 4.2. Example of the clipping processing method

Input text: [CLS] who is elmo elmo is a red muppet from sesame street who is bert bert is an nlp model does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text: 31

Input text after clipping: [CLS] who is elmo elmo is a red muppet from sesame street who is does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text after clipping: 25

- **BERT + summarization**: Ten runs with the respective number of conversation turns in the input. The input for BERT + summarization consists of n conversation turns concatenated with the query and a candidate passage. Whenever this input failed to meet the input restrictions, the context was summarized so that it fit. The processed example input is depicted in Figure 4.3.

Figure 4.3. Example of the summarization processing method

Input text: [CLS] who is elmo elmo is a red muppet from sesame street who is bert bert is an nlp model does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text: 31

Input text after summarization: [CLS] elmo bert sesame muppet nlp model does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text after summarization: 18

- **BERT + rank fusion AVG**: Ten runs with one of the conversation turns in the input. The input for BERT + rank fusion AVG consists of one conversation

turn concatenated with the query and a candidate passage. For each number of conversation turns, scores were obtained from each run, from which the average was calculated. These scores were transformed into rankings. The processed example input for all fusion methods is depicted in Figure 4.4.

- **BERT + rank fusion MAX**: Ten runs with one of the conversation turns in the input. The input for BERT + rank fusion MAX consists of one conversation turn concatenated with the query and a candidate passage. For each number of conversation turns, scores from each run were obtained, from which the maximum was selected. These scores were transformed into rankings.
- **BERT + rank fusion RRF**: Ten runs with one of the conversation turns in the input. The input for BERT + rank fusion RRF consists of one conversation turn concatenated with the query and a candidate passage. For each number of conversation turns, scores from each run were obtained, which were transformed into rankings. These rankings were fused using the formula discussed in Section 3.3.3.

Figure 4.4. Example of the rank and score fusion processing methods

Input text 1: [CLS] who is elmo elmo is a red muppet from sesame street does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text 1: 25

Input text 2: [CLS] who is bert bert is an nlp model does he know elmo [SEP] bert does not know elmo [SEP]
Number of tokens in input text 2: 20

4.2 Experiments on the TREC CAsT dataset

The experiments for TREC CAsT were setup similarly to those conducted for MS MARCO. However, because this is a much smaller dataset, no distinction was made between different numbers of conversation turns. The comparison was focused on all context or no context. Furthermore, there was no fixed query for the sessions: every query and its context was considered in all sessions. This resulted in a set of queries, of which some have no context and others have multiple previous conversation turns as context. Sessions that lacked a relevant passage from the MS MARCO collection were discarded. The model that we use was fine-tuned on MS MARCO data, so only the passages from that data were considered. This resulted in a test set consisting of 120 queries across 12 sessions. The same experiments were conducted: BM25, BERT + clipping, BERT + summarization, BERT + rank fusion AVG, BERT + rank fusion MAX and BERT + rank fusion RRF, but with two runs instead of ten.

4.3 Implementation details

This section describes how the techniques explained in the previous chapter were applied. The main coding language used is Python.

4.3.1 BM25

The Anserini toolkit [27] was employed to obtain candidate passages with BM25 and also to calculate the BM25 performance with more context. Anserini is an open-source information retrieval toolkit built on Lucene[1]. For the MS MARCO experiments, the MS MARCO passage collection was first indexed in Anserini, and for each query in the test set, 200 passages with the highest BM25 score were retrieved as the set of passages for re-ranking. As a second step, the obtained passages were used as the collection and indexed in Anserini. This made it possible to re-run BM25 with different inputs but with the same candidate passages for re-ranking.
For the TREC CAsT data, a small collection was indexed for each query in all sessions, consisting of the 20 candidate passages that were obtained from the relevance labels. These labels contain both positive and negative labels. For each session, BM25 was run with and without context.

4.3.2 BERT

An attempt was first made to run some experiments using the pre-trained but not fine-tuned version of BERT_{BASE}. This however did not result in a better performance than BM25. It was necessary to fine-tune BERT on the data used in this work, which was difficult given the available hardware; fine-tuning BERT would take multiple days on multiple *tensor processing units* (TPU's). However, it was possible utilize the work of Nogueira et al. [19] which we mentioned in Section 1.3.2. The authors used BERT as a passage re-ranker for the MS MARCO passage re-ranking task and achieved good results in comparison to BM25, KNRM [28] and Conv-KNRM [29]. The authors have published their code online[2] along with pre-trained BERT models that are fine-tuned on the MS MARCO question and answering data. Their BERT_{BASE} model is used in this work.
Nogueira et al. have implemented the model in TensorFlow[3]. This research, however, used PyTorch[4], and so it was necessary to convert the model weights from TensorFlow to PyTorch format. The Python library *Pytorch-Transformers*[5] provides a script for converting TensorFlow checkpoints to PyTorch checkpoints and was used to convert the weights. As Nogueira et al. defined their model as a fine-tuning model for sequence classification on top of the pre-trained BERT_{BASE}, a similar model in PyTorch had to be initialized before it was possible to load the now converted model weights. Calculating the BERT activations for the MS MARCO testset (582 times 200 candidate passages) for one experiment took about 2 hours on a GTX 1060 GPU. Calculating the activations for the TREC CAsT testset (120 times 20 candidate passages) for one experiment took about 15 minutes.

4.3.3 Summarization

For summarization, whenever the input was too long, only the context was summarized to avoid losing important information about the query to be answered and the candidate answer passage. In order to identify the most important terms in the context, a TD-IDF score had to be calculated for each term. A TD-IDF vectorizer,

[1] https://lucene.apache.org/ (Last accessed: 2/8/2019)
[2] https://github.com/nyu-dl/dl4marco-bert (Last accessed: 2/8/2019)
[3] https://www.tensorflow.org/ (Last accessed: 2/8/2019)
[4] https://pytorch.org/ (Last accessed: 2/8/2019)
[5] https://github.com/huggingface/pytorch-transformers (Last accessed: 5/8/2019)

implemented in the Python package *scikit learn*[6], was fitted on the MS MARCO passage collection. This fitted vectorizer could then be used to transform the context and obtain the TF-IDF scores from it, from which the top n terms were selected. The number of turns (n) was relative to the total number of words in the context; only 30% of the words were selected. This was done because in this form of summarization, a bag-of-words representation of the context is obtained, and all sentence structure is lost. Selecting only 30% of the top terms ensures that only meaningful words are obtained. This percentage yielded the best results.

4.3.4 Rank and score fusion

Instead of concatenating the previous utterances, n separate inputs were defined for each session as follows: previous conversation turn + query + candidate passage. n is the number of conversation turns. The obtained ranks were then combined using the AVG, MAX and RRF methods described in Section 3.3.3.

4.4 Evaluation metrics

The metrics that are proposed in CAsT are Precision @ 1 (P@1), Precision @ 3 (P@3), Expected Reciprocal Rank (ERR) and mean Average Precision (mAP). The experiments in this work are evaluated with P@1, P@3, Mean Reciprocal Rank (MRR) instead of ERR and additionally the mean rank. The mean rank is a very simple metric, which denotes the mean of the rank of all positive labels.

Because of the nature of the task in this research, the highest ranked result is the main interest; in a conversational setting, one cannot keep giving answers until the right one is found, there is only one opportunity to answer. mAP was not used because this metric does not necessarily provide information on the highest ranked result and due to other weaknesses that are described by Fuhr [30] in a study on common mistakes in IR evaluation. Fuhr also mentions shortcomings of ERR and MRR. However, MRR was still used in this work and Section 4.4.2 describes the reasoning behind this.

4.4.1 Precision @ k

$P@k$ is a very simple retrieval metric that corresponds to the number of relevant items in the first k results. This metric has some obvious flaws: firstly, it does not take the actual position in the ranking into account. Secondly, if a query has fewer relevant results than k, even a perfect retrieval will result in a score that is lower than one. It is however still a useful metric for this work because as we mentioned before, the highest rank is the main interest and this metric can provide insights for that.

[6]https://scikit-learn.org/ (Last accessed: 13/6/2019)

4.4.2 Mean Reciprocal Rank

The regular Reciprocal Rank (RR) calculates the multiplicative inverse of the rank at which the first relevant result was retrieved, for example: 1 for first place, $\frac{1}{2}$ for second place, $\frac{1}{3}$ for third place etc. The MRR is the average of the reciprocal ranks of all queries. The formula for the MRR is provided in Equation 4.1:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{r_i} \tag{4.1}$$

Where Q is a set of queries and $|Q|$ denotes the total number of queries. r_i refers to the rank of the first relevant result for the i-th query.

Fuhr [30] describes the shortcomings of this metric and ERR, which is an adaptation of the MRR. For both metrics, having two relevant results at rank one and three gives a better score than having two relevant results at rank two, which is 'strange behavior' according to Fuhr. Even though this might be unwanted behavior in general, this is not the case for the task in this study, where it is useful since the focus lies on the highest ranked result. Retrieving a relevant result at rank 100 is as equally useless as retrieving it at rank 200. Therefore, MRR was used as an evaluation metric.

Chapter 5

Results

In the first section of this chapter, the results obtained in the experiments on both the MS MARCO and the TREC CAsT datasets are presented. Thereafter, an interpretation of those results is described.

5.1 Results on the MS MARCO dataset

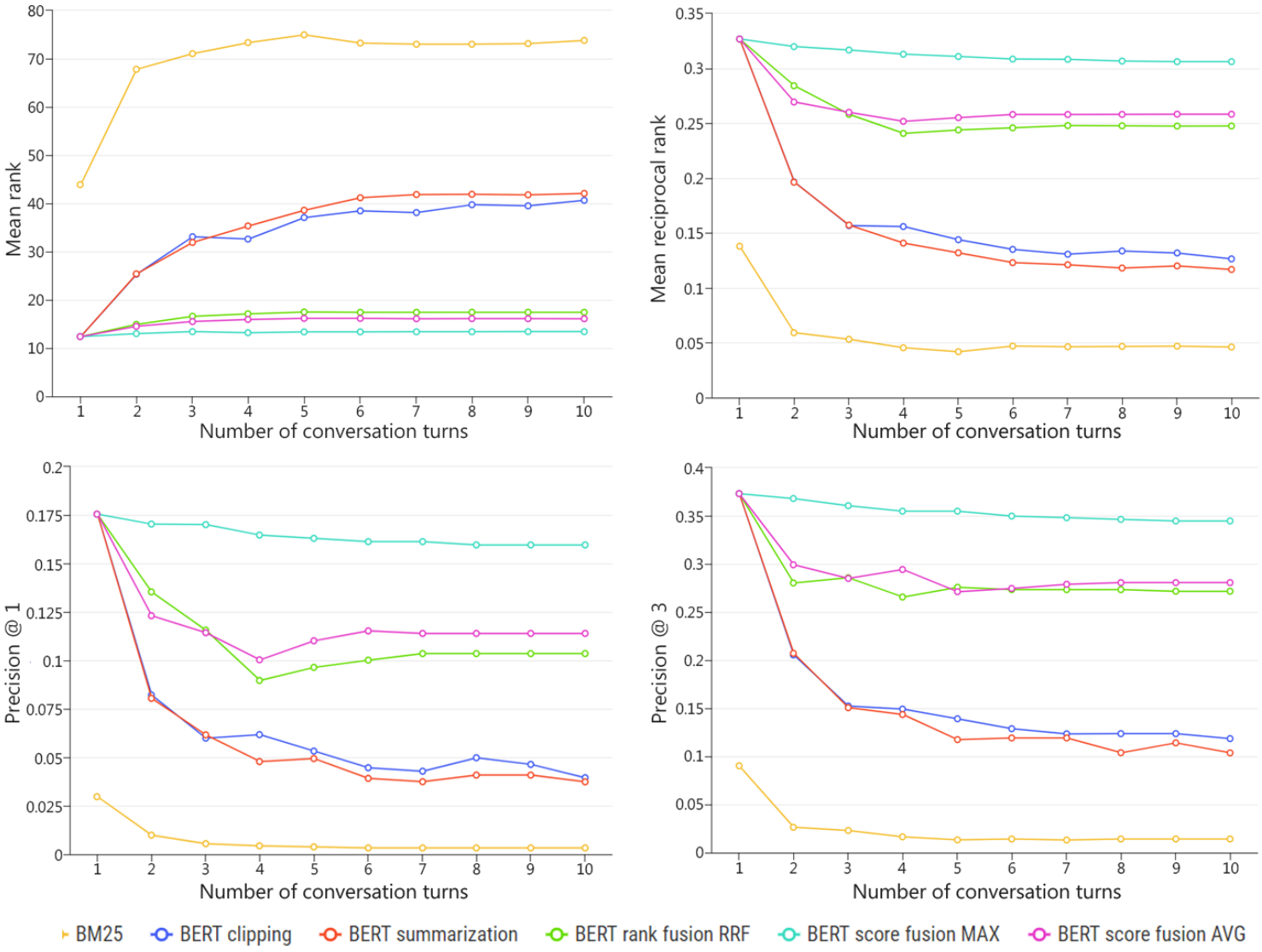

FIGURE 5.1: MS MARCO dataset results

Figure 5.1 shows the results of the experiments conducted on the test set created from the MS MARCO artificial conversational sessions dataset.

5.2 Results on the TREC CAsT dataset

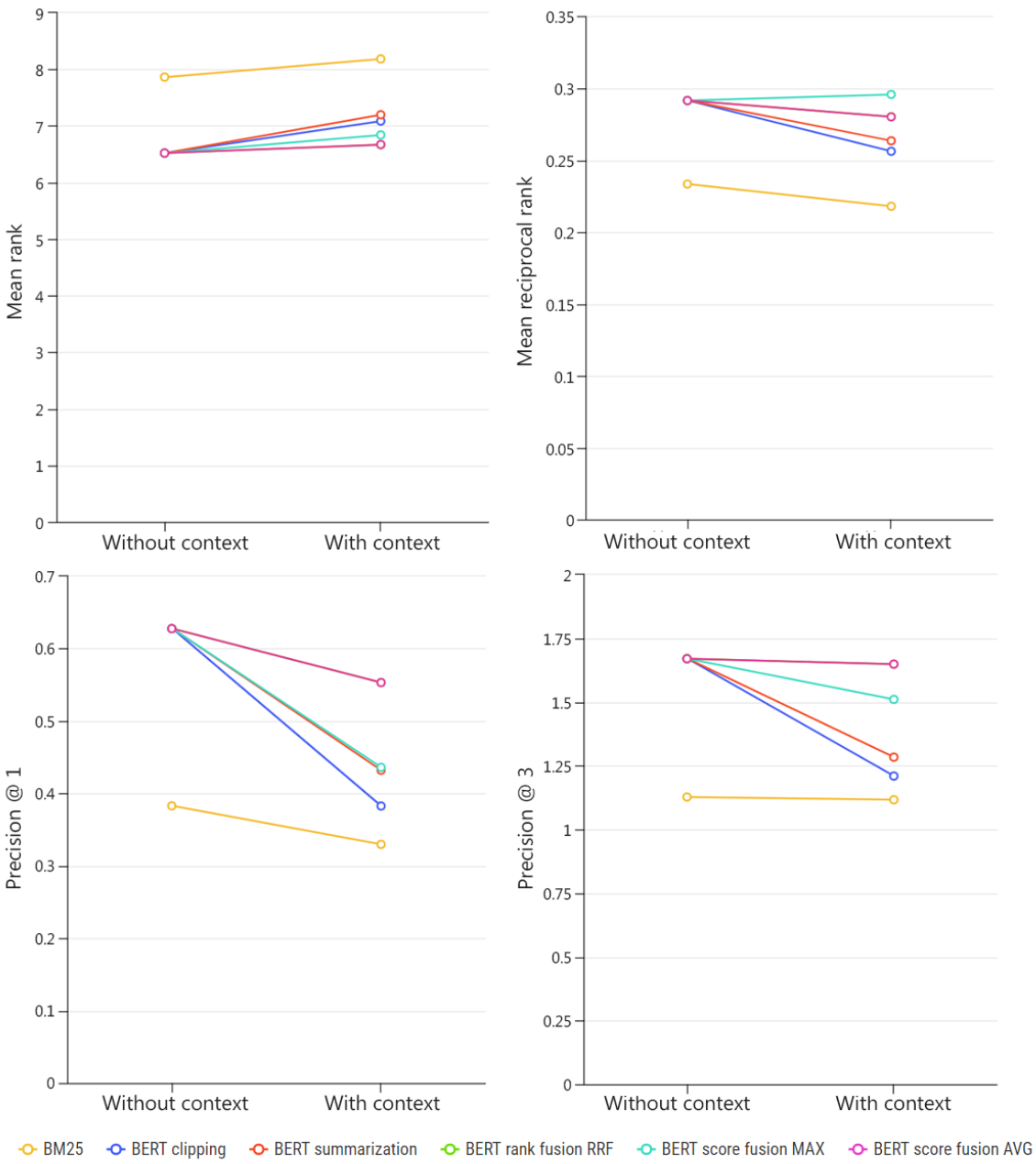

FIGURE 5.2: TREC CAsT dataset results

Figure 5.2 shows the results of the experiments conducted on the test set created from the TREC CAsT conversational dataset.

5.3 Evaluation

This section elaborates on the results presented in the figures above.

5.3.1 MS MARCO

Figure 5.1 presents four plots illustrating the results using the four metrics discussed in Section 4.4. Each line represents one experiment, and this is tracked for ten runs. Each run was calculated with a different number of conversation turns as context in the input.

For all experiments, the best predictive performance was achieved in this first run, without any context having been added. Overall, BERT outperformed BM25 by a large margin. Within the BERT context processing experiments, even though it was not useful to add context, the performance of the different methods can still be compared.

The best scores were achieved with the score and rank fusion methods and the MAX method in particular. A possible explanation for these results is described below.
Starting with the summarization experiment; if the context is summarized in an extractive manner, all of the linguistic structure contained in the sentences is lost, while the most important words are retained. It can be argued that summarization does not work well because BERT is trained to make sense of grammatically correct sentences. Also, the words that have the highest TF-IDF score in relation to the MS MARCO passage collection, might not be the words that have predictive value over the candidate passages.
As for the rank and score fusion, it is observed that the highest scores were achieved with the MAX method. In many sessions, not all context is relevant for answering the query. In all other experiments besides the MAX method, all possible context utterances were incorporated even though they might be irrelevant to the candidate passage, which could have led to lower scores. However, when the MAX method was used to fuse the scores, only the context utterances with the highest activations were considered. This could explain why MAX fusion performs relatively well.

5.3.2 TREC CAsT

The results for TREC CAsT are setup in a similar manner to MS MARCO and are depicted in Figure 5.1. However, there were only two runs for these experiments, where the context was either considered or it was not. The best predictive performance for all experiments was achieved without adding any context. Similarly to the MS MARCO results, BERT outperformed BM25, and it performed best when processing the input with any fusion method.

Chapter 6

Discussion

In this chapter, the findings described in the previous chapter are discussed in light of the research problem. The results for the MS MARCO and the TREC CAsT datasets are interpreted and discussed separately. Both datasets proved sub-optimal for the experiments in this study and their drawbacks are explained. Subsequently, other limitations of this study are discussed and future work is suggested.

6.1 MS MARCO

The results of the experiments on the MS MARCO dataset indicate that adding context to the input is not useful for answering a query. This could partly be due to the limitations of the context processing methods described in Section 5.3.1, but quality of the dataset is also addressed as a point of discussion. The MS MARCO conversational dataset consists of synthetic sessions in which the queries were clustered together in an unsupervised manner. It can be assumed that the queries in a session share at least one word or have a common subject. However, this does not imply that the queries in a session and their answers are sufficiently relevant that they help answering a similar query. The official task that was designed for this dataset might also be an indication that the queries do not carry predictive information on answering other queries in a session, the task was described as follows: 'Given a session with 2-n queries with one query being masked, predict the masked query'.

6.2 TREC CAsT

Before it was decided to run the experiments with the positive and negative relevance labels provided by TREC CAsT, the experiments were run with passages retrieved from the MS MARCO passage collection. Similarly to the MS MARCO experiments, 200 passages were retrieved using BM25, and the number of positive labels among them was determined. Of the total number of 515 positive labels across 120 queries, without adding context, 250 positive labels were retrieved. When context was added to the queries, 265 positive labels were retrieved, only 15 more. This indicates that even with a small number of retrieved passages and a baseline ranker such as BM25, the context does not carry much information.

Eventually the final experiments were run with the positive and negative labels provided by TREC CAsT because with this approach, the full set of 120 queries spread across 12 sessions could be utilized. This was important because the usable section of the dataset for this work was already very small.

6.3 Future work

The amount of data that was used to evaluate the MS MARCO dataset was chosen based on a trade-off between using a representative sample of the data and the time needed to run the experiments. It can be argued that this sample was a bad representation of the conversational data, the experiments could be repeated with more data in order to confirm this. As for the TREC CAsT data, the size of the set can also be an indication that the results are not representative of a realistic performance and these experiments could be repeated for comparison once all the queries across the 80 sessions have received relevance labels.

This research was conducted in the context of the TREC conversational assistance track, and the information that has been gathered can be used to determine the approach to be used in our submission for the challenge. Although the results seem to indicate that it is not useful to use the context in prediction, there are other possible modifications that can be tried that might be useful. Firstly, BERT can be fine-tuned on conversational data instead of just question answering data, preferably on realistic conversations and not artificially generated ones. It might also be useful to fine-tune BERT using other data sources besides MS MARCO to improve the generalizability of the model.

Also, the comparison between the different context processing methods indicates that there is at least some predictive information in the context, otherwise the performance of all methods would have been similar. However, the experiments described in this work have been unsuccessful in separating the noise from the useful content. Other context processing experiments could be conducted to see whether the useful content can be isolated from the noise. For example, before adding the context to the query to be used as input, the context can be analyzed by an NLP model to examine whether this context has any predictive information on the candidate passage.

Lastly, ever since the experiments in this work were established, other models in the NLP domain have been released that outperform BERT on multiple NLP tasks, two of these models are described below.
Yang et al. [31] propose a network called XLNet, which the authors describe as a generalized autoregressive pretraining method. XLNet is pre-trained using ten times the amount of data that BERT_{base} uses. It also uses a batch size eight times larger for half as many optimization steps, which allows it to process four times as many sequences in the pre-training step. BERT and XLNet employ different approaches to create the representations in a bidirectional way. XLNet outperforms BERT on 20 tasks such as reading comprehension, text classification and the GLUE tasks[1]. However, despite the good performance, the computational requirements of XLNet are much higher than those of BERT. Given the time and resources needed for the experiments in this research, using XLNet could only be feasible if there is enough time and computational power available. Liu et al. propose [32] the Multi-Task Deep Neural Network (MT-DNN) that outperforms BERT in eight out of nine GLUE tasks[1] and other NLP benchmarks. MT-DNN extends the representation learning model proposed by Liu et al. [33] by incorporating BERT in the architecture.
One could experiment with these models on a small sample of the dataset to see whether they can improve predictive performance of the context.

[1] https://gluebenchmark.com/tasks (Last accessed: 5/8/2019)

Chapter 7

Conclusion

In this work, experiments were conducted for the task of text passage retrieval and re-ranking for question answering, given conversational context. Two models, BERT and BM25 were compared, and it was researched how including conversational context in the input influences the predictive power of these models. Multiple methods for dealing with the input restrictions of BERT were evaluated. Furthermore, the study compared experiments on two conversational datasets that were constructed in different manners: either artificially, with clustering methods, or by hand. Four research questions were formulated, which are answered below.

- **RQ1** - *How does BERT perform in a conversational re-ranking task in comparison with the baseline ranker BM25?*

  BERT outperforms BM25 in all of the experiments conducted in this work, regardless if or how much context is taken into account.

- **RQ1.1** - *To what extent does predictive power change when adding conversational context to the input?*

  In the experiments presented here, adding context to the input does not increase predictive power for BERT or BM25.

- **RQ1.2** - *How can BERT's input restrictions (512 tokens) be dealt with when using conversational data?*

  Five different methods were employed to deal with the input restrictions: clipping as a baseline, summarization, maximum score fusion, average score fusion and reciprocal rank fusion. The fusion methods outperform the clipping and summarization methods.

- **RQ2** - *How does the value of context differ across an artificially generated conversation dataset and a more realistic conversational dataset?*

  In this study, adding context to the inputs from the artificially generated dataset and the conversational dataset is equally unbeneficial.

Bibliography

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding", *ArXiv preprint arXiv: 1810.04805*, 2018.

[2] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, *et al.*, "Okapi at TREC-3", *NIST Special Publication SP*, vol. 109, p. 109, 1995.

[3] J. Weizenbaum, "Eliza—a computer program for the study of natural language communication between man and machine", *Communications of the ACM*, vol. 26, no. 1, pp. 23–28, 1983.

[4] A. M. Turing, "Computing machinery and intelligence", *Mind*, vol. 49, no. 236, pp. 433–460, 1950.

[5] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: An automatic question-answerer", in *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, ACM, 1961, pp. 219–224.

[6] E. M. Voorhees *et al.*, "The TREC-8 question answering track report", in *TREC*, Citeseer, vol. 99, 1999, pp. 77–82.

[7] Z. Ji, Z. Lu, and H. Li, "An information retrieval approach to short text conversation", *ArXiv preprint arXiv:1408.6988*, 2014.

[8] J. L. Elman, "Finding structure in time", *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] M. Tan, C. d. Santos, B. Xiang, and B. Zhou, "Lstm-based deep learning models for non-factoid answer selection", *ArXiv preprint arXiv:1511.04108*, 2015.

[11] Y. Shen, Y. Deng, M. Yang, Y. Li, N. Du, W. Fan, and K. Lei, "Knowledge-aware attentive neural network for ranking question answer pairs", in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ACM, 2018, pp. 901–904.

[12] B. Wang, K. Liu, and J. Zhao, "Inner attention based recurrent neural networks for answer selection", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 1288–1297.

[13] Y. Wu, Z. Li, W. Wu, and M. Zhou, "Response selection with topic clues for retrieval-based chatbots", *Neurocomputing*, vol. 316, pp. 251–261, 2018.

[14] L. Yang, Q. Ai, J. Guo, and W. B. Croft, "Anmm: Ranking short answer texts with attention-based neural matching model", in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM, 2016, pp. 287–296.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need", in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[16] X. Zhou, L. Li, D. Dong, Y. Liu, Y. Chen, W. X. Zhao, D. Yu, and H. Wu, "Multi-turn response selection for chatbots with deep attention matching network", in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2018, pp. 1118–1127.

[17] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations", *ArXiv preprint arXiv: 1802.05365*, 2018.

[18] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training", *URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf*, 2018.

[19] R. Nogueira and K. Cho, "Passage re-ranking with BERT", *ArXiv preprint arXiv:1901.04085*, 2019.

[20] R. Lowe, N. Pow, I. Serban, and J. Pineau, "The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems", *ArXiv preprint arXiv:1506.08909*, 2015.

[21] Z. Zhang, J. Li, P. Zhu, H. Zhao, and G. Liu, "Modeling multi-turn conversation with deep utterance aggregation", *ArXiv preprint arXiv:1806.09102*, 2018.

[22] R. Yan, Y. Song, and H. Wu, "Learning to respond with deep neural networks for retrieval-based human-computer conversation system", in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, 2016, pp. 55–64.

[23] H. Padigela, H. Zamani, and W. B. Croft, "Investigating the successes and failures of BERT for passage re-ranking", *ArXiv preprint arXiv:1905.01758*, 2019.

[24] G. Salton and M. J. McGill, "Introduction to modern information retrieval", 1986.

[25] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. 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*, 2015, pp. 19–27.

[26] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods.", in *SIGIR*, vol. 9, 2009, pp. 758–759.

[27] P. Yang, H. Fang, and J. Lin, "Anserini: Reproducible ranking baselines using lucene", *Journal of Data and Information Quality (JDIQ)*, vol. 10, no. 4, p. 16, 2018.

[28] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling", in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, 2017, pp. 55–64.

[29] Z. Dai, C. Xiong, J. Callan, and Z. 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*, ACM, 2018, pp. 126–134.

[30] N. Fuhr, "Some common mistakes in IR evaluation, and how they can be avoided", in *ACM SIGIR Forum*, ACM, vol. 51, 2018, pp. 32–41.

[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding", *ArXiv preprint arXiv:1906.08237*, 2019.

[32] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding", *ArXiv preprint arXiv:1901.11504*, 2019.

[33] X. Liu, J. Gao, X. He, L. Deng, K. Duh, and Y.-Y. Wang, "Representation learning using multi-task deep neural networks for semantic classification and information retrieval", 2015.