DESIGN AND IMPLEMENTATION OF A RESPONSE RETRIEVAL AND RERANKING SYSTEM

by

Mustafa Can Deveci

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical & Electronics Engineering
Boğaziçi University

2022

ACKNOWLEDGEMENTS

I would like to present my special thanks to Prof. Murat Saraçlar for being such a great advisor for my master's thesis. He inspired me as an example of perfect academician and has always been a role model for me.

Many thanks to Assoc. Prof. Arzucan Özgür and Assist. Prof. Ebru Arısoy-Saraçlar for their participations in my thesis committee and for their contributions.

I would like to mention Assoc. Prof. Bülent Altunkaya, my high school Mathematics teacher who planted the seeds of the love for Mathematics in me. A real modern-time Anaximander.

I would also like to present my sincere thanks to my lovely company and home, Spotzer Digital. Matthew Christian, Gerçek İlke Gültekin, and Bertram Croes; thanks for your endless understanding and patience throughout the year of my master's thesis and being the greatest team in the known universe.

ABSTRACT

DESIGN AND IMPLEMENTATION OF A RESPONSE RETRIEVAL AND RERANKING SYSTEM

In this master's thesis, we started with a baseline response retrieval and re-ranking system that is composed of two steps: BM25 retrieval and BERT re-ranking. After investigating the effects of several parameters and BERT model size on the baseline approach, a novel retrieval and re-ranking system with TF-IDF retrieval and Cross-Encoder re-ranking steps was designed and implemented. With the application of Deep Learning models to the re-ranking step, consistent ranking performance improvements have been observed. The research focus of this thesis is a comparative performance study of different Transformer models. In the experiments carried on in this thesis, we showed that smaller transformer models can out-perform larger models. Additionally, this designed re-ranking system was re-purposed for a Question Answering task where the answer for a given question is searched as a subset of a passage. Even though the re-ranking system was directly used without undergoing any modifications regarding the QA task, promising results that are worth further research have been attained.

ÖZET

YANIT EDİNİMİ VE SIRALAMASINA DAYALI SİSTEM TASARIMI VE GERÇEKLENMESİ

Bu yüksek lisans tezinde, çalışmaya BM25 yanıt edinimi ve BERT Transformer modeli bazlı yeniden sıralama adımlarından oluşan genel bir yanıt alma ve yeniden sıralama sistemiyle başlanmıştır. Bu baz yaklaşımının üzerinde birtakım genel parametre değişikliklerinin ve BERT model boyutunun sistem performansı üzerindeki etkileri araştırıldıktan sonra, TF-IDF yanıt edinimi ve Cross-Encoder yeniden sıralama yaklaşımlarına dayalı özgün bir yanıt edinim ve yeniden sıralama sistemi tasarlanmış ve gerçeklenmiştir. Derin Öğrenme modellerinin yeniden sıralama adımına uygulanmasıyla tutarlı bir sıralama performansı artışı gözlemlenmiştir. Bu tezde, araştırma konusu odağı olarak çeşitli Transformer modellerinin karşılaştırmalı başarım çalışması seçilmiştir. Yapılan deneylerde küçük Transformer modellerden nispeten daha büyük modellere göre daha iyi performans alınabileceği gösterilmiştir. Son olarak, bu yanıt sıralama sistemi, belirli bir sorunun cevabının bir pasajın alt kümesi olarak arandığı Soru Cevaplama görevi için yeniden kullanılmıştır. Sistem üzerinde Soru Yanıtlama görevine yönelik herhangi bir modifikasyon yapılmadan yanıt sıralama yaklaşımı doğrud an kullanılmasına rağmen, araştırmaya açık ve umut vadeden sonuçlar elde edilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . ix
LIST OF SYMBOLS . . . xi
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
1.1. Conversational Systems Taxonomy . . . 2
1.1.1. Interaction Mode . . . 2
1.1.2. Task-Orientedness . . . 3
1.1.3. Knowledge Domain . . . 3
1.1.4. Design Techniques . . . 4
1.2. Research Focus and Main Contributions . . . 4
2. LITERATURE REVIEW . . . 6
2.1. Datasets and Applications of Neural Networks for Information Retrieval 6
2.2. Transformers and BERT . . . 7
2.3. Semantic Search . . . 8
2.4. Sentence Transformers . . . 9
3. DATASETS . . . 12
3.1. MS MARCO . . . 12
3.2. TREC CAR . . . 15
3.3. Lecture Question Answering Dataset . . . 17
4. METHODS . . . 20
4.1. Baseline Implementation . . . 20
4.2. Experiments on Lecture QA Dataset . . . 23
4.2.1. TF-IDF Retrieval Phase . . . 24
4.2.1.1. Preprocessing . . . 24

4.2.1.2. Calculation of TF-IDFs . . . 25
4.2.1.3. Retrieve Docs for Each Question . . . 26
4.2.2. Re-ranking . . . 27
4.2.2.1. SentenceTransformers . . . 27
4.2.2.2. Pre-trained Cross-Encoders Models As Rerankers . . . 33
4.2.2.3. Custom Fine-tuning with Huggingface . . . 36
4.2.2.4. Further Fine-tuning with Lecture QA Dataset . . . 37
4.3. Question Answering Experiments . . . 38
5. EVALUATION . . . 39
5.1. Baseline Experiments . . . 39
5.1.1. MAP . . . 39
5.1.2. MRR . . . 39
5.2. Re-ranking . . . 40
5.2.1. NDCG . . . 40
5.2.2. MAP and MRR in Re-ranking Experiments . . . 40
5.2.3. Catch @n . . . 41
5.3. Question Answering . . . 41
5.3.1. SQuAD Official Evaluation Script . . . 41
5.3.2. Rouge . . . 41
5.3.3. BLEU . . . 42
6. RESULTS AND DISCUSSION . . . 43
6.1. Baseline Approach . . . 43
6.2. Re-ranking Results . . . 46
6.3. Question Answering Results . . . 51
7. FUTURE WORK . . . 57
8. CONCLUSION . . . 59
REFERENCES . . . 60
APPENDIX A: PERMISSIONS ABOUT FIGURE USAGE . . . 68

LIST OF FIGURES

Figure 4.1. A Typical Information Retrieval System. 21

Figure 4.2. Preprocessing Steps. 26

Figure 4.3. Basic Training Architecture [1]. 30

Figure 4.4. Bi-Encoder architecture with softmax classifier objective function [2]. 31

Figure 4.5. Bi-Encoder architecture with cosine similarity objective function [2]. 32

Figure 4.6. Cross-Encoder Architecture [3]. 33

Figure 6.1. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, m-NDCG results. 52

Figure 6.2. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, MAP results. 52

Figure 6.3. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, catch@n results. 53

Figure 6.4. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, Model Sizes. 53

Figure A.1. Received permission about the usage of Figures 4.3, 4.4, 4.5, and 4.6. 69

LIST OF TABLES

Table 2.1. BERT models and their sizes. 7

Table 3.1. Samples from the MS Marco Train Triples Small [4]. 14

Table 3.2. A sample set from Lectures QA Dataset [5]. 18

Table 3.3. Lecture QA Dataset Statistics [5]. 18

Table 4.1. Removed Punctuation Marks and Symbols. 24

Table 4.2. Released Fine-tuned Model Performances on MS MARCO, runtimes were computed on a V100 GPU using Huggingface Transformers v4 [2]. 35

Table 6.1. Baseline Experiments on TREC CAR Dataset. 44

Table 6.2. Model Size Experiments on TREC CAR Dataset. 45

Table 6.3. Re-ranking Results Mean-NDCG and MAP Evaluations. 47

Table 6.4. Re-ranking Results catch@n Evaluations. 50

Table 6.5. Retrieval and Re-Ranking performance of further-fine-tuning-modified model. 51

Table 6.6. Question Answering Task SQuAD F1 Evaluation. 54

Table 6.7. Question Answering Task Rouge Evaluation. 55

Table 6.8. Question Answering Task BLEU Evaluation. 55

Table 6.9. Sample 1 from Lecture QA Dataset (eval) [5]. 56

Table 6.10. Sample 2 from Lecture QA Dataset (eval) [5]. 56

Table 7.1. A sample from Lecture QA Dataset (eval) including domain knowledge [5]. 58

LIST OF SYMBOLS

J_{neg}	Indices of non-relevant passages
J_{pos}	Indices of relevant passages
L	Loss
p_i	An item of a retrieved collection of passages
q	Query
s_i	Relevance score of a passage
$\overrightarrow{u}$	Vectoral representation of a sentence

LIST OF ACRONYMS/ABBREVIATIONS

AI	Artificial Intelligence
AP	Average Precision
BERT	Bidirectional Encoder Representations from Transformers
BLEU	Bilingual Evaluation Study
CAR	Complex Answer Retrieval
CG	Cumulative Gain
CI	Conversational Interface
DCG	Discounted Cumulative Gain
DF	Document Frequency
GPU	Graphical Processing Unit
GUI	Graphical User Interface
IDF	Inverse Document Frequency
IR	Information Retrieval
LCS	Longest Common Subsequence
MAP	Mean Average Precision
MIT	Massachusetts Institute of Technology
MLM	Masked Language Modeling
MRR	Mean Reciprocal Rank
NDCG	Normalized Discounted Cumulative Gain
NLI	Natural Language Inference
NLP	Natural Language Processing
NLU	Natural Language Understanding
NSP	Next Sentence Prediction
OCW	OpenCourseWare
PC	Personal Computer
QA	Question Answering
RAM	Random Access Memory
ROC	Region of Convergence

SBERT	Sentence-BERT
Seq2Seq	Sequence-to-Sequence
SR	Speech Recognition
STS	Semantic Text Similarity
TF	Term Frequency
TF-IDF	Term Frequency Inverse Document Frequency
TREC	Text REtrieval Conference
TSV	Tab Separated Values
TTS	Text-to-Speech
USE	Universal Sentence Encoder
XML	Extensible Markup Language

1. INTRODUCTION

One of the major aims in the the research field of Artificial Intelligence (AI) is to build machines that can converse with us and respond to queries or questions mimicking humans [6]. This goal has been historically a deep desire for researchers in the field of conversational AI. One of the most iconic works was suggested by the great computer scientist and one of the founders of artificial intelligence, Alan Turing. Turing test can be considered "passed" if a computational agent (a device) can perform on the same level with a human being so that it is capable of tricking a human judge into believing that it is indeed a human being.

Beyond being an intriguing academic research area, conversational systems (typically using a response retrieval and re-ranking scheme) have evolved to a level of maturity so that such systems started finding application areas in industrial solutions. Commercial companies utilize conversational AI to develop communication channels with customers because AI can relieve the burden of labor in customer services. Everyday, customer service departments of companies have to face the same questions and requests, repeatedly. This massive and repetitive workload on manpower can be transferred to autonomous systems in an efficient and effective manner via the usage of conversational agents. This trend is, indeed, what we have been observing in the interactions between companies and their customers. For simple tasks, it is convenient to use these Conversational Interfaces (CI) instead of sparing huge amount of human resource for customer relations or using a Graphical User Interface (GUI) leading to the burden of learning this GUI.

Another example for application areas of conversational interface is personalized assistants for personal computers (PC) and other devices such as smartphones or tablets. This trend seems to be continuing because of the improvements in the abilities of such systems. Using these types of agents are getting easier day-by-day. Siri by Apple, Cortana by Microsoft, Alexa by Amazon, and Google Assistant can be given as

the most renowned examples for such systems.

1.1. Conversational Systems Taxonomy

The domain of conversational systems can be divided into sub-categories according to a number of criteria such as interaction mode, task-orientedness, knowledge domain, and design techniques [7]. In this section, a coarse grained map of the dialogue systems domain is presented to locate the subject of this thesis in the domain of conversational or dialogue systems.

1.1.1. Interaction Mode

To facilitate the interaction of a user with a computer, conversational agents utilizes Natural Language Processing (NLP), and Natural Language Understanding (NLU) techniques to capture the semantics of an utterance which is either written or uttered by a human. Therefore, conversational user interfaces can be divided into two sub-categories: text-based and voice-based interfaces. The first example that comes to mind for text-based assistants would be chatbots. Today, chatbots are predominantly used in mobile applications and webpages of companies (e-commerce sites, financial institutions, and telecom companies etc.) having voluminous and repetitive communication with customers. On the other side, voice assistants help users interact with a system simply by uttering a query to the system. Siri by Apple, Cortana by Microsoft, Alexa by Amazon, or Google Assistant are well-known examples of voice assistants. Voice assistants are usually text-based assistants with Speech Recognition (SR) and Text-to-Speech (TTS) layers surrounding the text-based chatbot system. This thesis implements a response retrieval and re-ranking system as the core conversational system that can be combined with SR and TTS modules.

1.1.2. Task-Orientedness

Another taxonomy can be made according to task-orientedness (goal-drivenness) of dialog systems [6]. Task-oriented systems are shaped around a pre-defined task such as technical support issues, booking systems, and querying systems. These types of systems have strict or nearly-strict dialogue flows. For instance; when a customer calls a voice-based assistant of a bank, it would be expected that there are certain goals of this call such as questioning remaining credits in an account, paying a bill, or de-activating a lost credit card. On the other hand, we have non-task-driven systems which are also referred to as chit-chat models. Typically, there is no ultimate end-point for communicating with these agents. These systems are designed for short and unstructured conversations. However, these systems are good indicators for state-of-the-art methods for modeling human-like conversations. Since there is no pre-defined intention for the query-response system implemented in this thesis, we can consider this work as a non-task-oriented system. Compared to task-oriented dialog systems, building open-ended non-task-oriented conversational systems is more challenging.

1.1.3. Knowledge Domain

The domain of knowledge used in a conversational system is another criterion for classification of such systems. Open-domain and closed-domain is the taxonomy with respect to the knowledge domain. Open domain dialog systems use a wide range of knowledge bases for their workings. Non-task-oriented systems where there is no contextual depth of conversation are usually open domain models. Conversely, closed-domain dialog systems are usually task-oriented systems using a more limited variety in source of information. Here, the knowledge domains of the systems in this thesis depend on the experiment sets implemented as will be seen in Methods chapter of this report.

1.1.4. Design Techniques

There are several approaches to design a dialog system. These approaches may vary for different types of systems. There are three basic design approaches for conversational systems: rule-based, retrieval-based, and generative-based approaches [7]. All these approaches have a number of solutions to be utilized while returning an answer for an utterance of a user. To exemplify, rule-based approaches generally uses pattern matching algorithms, retrieval-based systems can use BM25 [8] algorithm for retrieval, and generative approaches may have Sequence-to-Sequence (Seq2Seq) models to generate relevant responses for a query or a question.

1.2. Research Focus and Main Contributions

In this thesis, we start by studying a baseline passage re-ranking problem. However, this baseline implementation uses Tensorflow 1 framework [9] which is getting out-dated more on a daily basis with more modern deep learning frameworks emerging such as Tensorflow 2 and PyTorch. Therefore, we, first, re-write an end-to-end IR pipeline from scratch and utilize PyTorch giving us the ability to easily try new models and transformer architectures through Huggingface [10]. We also pick a new dataset which was originally intended as a Question Answering (QA) dataset. Second, we re-purpose this new dataset as a passage re-ranking dataset and test our approaches on it. Therefore, we have been able to make a comparative study of transformer models on re-ranking performance which is the main contribution of this work. And third, we also re-purpose our passage re-ranking approach as a Question Answering pipeline and evaluate its performance on the previously mentioned QA dataset.

In Chapter 2, we give the literature review in the field. In Chapter 3, we explain the datasets used in this work; in Chapter 4, we outline newly implemented methods in this work. In Chapter 5, we present the evaluation metrics and their computation methods; in Chapter 6, we present and discuss the results obtained with our models and in Chapter 7, we review possible research areas based on this work as future work. We

conclude this thesis report in Chapter 8 with the summary of the research conducted and the observations made in this work.

2. LITERATURE REVIEW

In this chapter, we review the existing publications in the literature. We start by mentioning datasets and implementations on these datasets. And then, we survey Transformer models and other paradigms in the field of passage retrieval and re-ranking systems.

2.1. Datasets and Applications of Neural Networks for Information Retrieval

With the addition of the new larger and more realistic datasets into the research field Information Retrieval (IR), neural network architectures have been proposed, studied, and proven to be working. Such as DRMM [11], KNRM [12], Co-PACRR [13], and DUET [14].

However, specifically speaking for the field of passage re-ranking under IR, this hasn't been the case. The datasets proposed are relatively new and still in need for further research. As a result of this limited amount of datasets to train neural models on, the use of neural architectures lagged behind. However, TREC-CAR was one of the first and well-researched large-enough datasets in the literature [15].

MS MARCO dataset made it possible to make experiments with neural architectures and research in the domain of machine comprehension [4]. These two datasets will be explained further in the following Datasets chapter.

On these two datasets, BERT models mostly have been parts of the most successful approaches to the passage re-ranking problem [16]. Plus, BERT-like models such as RoBERTa, ALBERT [17], and Electra [18] even surpassed the performance of BERT models for several tasks.

2.2. Transformers and BERT

BERT [19] is a pre-trained transformer neural network [20] example. Since its release, BERT has determined the state-of-the-art results for various NLP tasks including Question Answering, Sentence Classification, Text Summarization, Machine Translation, and Sentence-Pair Regression.

BERT (Bidirectional Encoder Representations from Transformers) [19] is a successful transformer implementation that has achieved state-of-the-art results for many NLP tasks. It employs transfer learning paradigm such that it is pretrained on a large and unlabeled dataset (Wikipedia) and fine-tuned on a down-stream dataset (MS Marco and/or TREC CAR).

There are 5 different default sizes of BERT models as shown in Table 2.1. The main difference between these models is the number of parameters.

Table 2.1. BERT models and their sizes.

Model Name	**Layers**	**Hidden Size**
BERT-Large	24	1024
BERT-Base	12	768
BERT-Medium	8	512
BERT-Mini	4	256
BERT-Tiny	2	128

As with the usage of transformer networks, the size of the models used in the research field has been getting larger. One reason for this is that with the increasing number of trainable parameters in the model, the capacity of the model increases. As the capacities of these models increase, researchers utilize larger and larger corpora to pre-train and fine-tune their models in the transfer learning paradigm. However, this is not a sustainable strategy because larger models out-grew memory sizes of computational units. They started not to fit into Random Access Memories (RAMs) of Graphical Processing Units (GPUs) that they are trained with. Therefore, smaller

models with more efficient pre-training, fine-tuning and evaluation implementations have emerged.

Lan et. al. present two parameter reduction methods to decrease memory consumption and speed up the training process [17]. Comparative experiments have shown that this strategy helps the model scale better than BERT models. They also used a self-supervised loss focusing on inter-sentence coherence modeling. They achieve state-of-the-art performances with their models on several NLP challenges.

Sanh et al. proposed an approach for pre-training a smaller version of generic purpose transformer model called DistilBERT [21]. They proved that knowledge distillation while pre-training of transformer models can decrease the size of a standard BERT model even though preserving the model performance. They achieved this by utilizing a triple loss comprised of language modeling, distillation, and cosine-distance losses.

Clark et al. presented a pre-training task that is performing better in terms of sample efficiency [18]. They manipulated the standard pre-training strategy used in BERT. Instead of masking some words and trying to predict the masked words, Clark et al. replaced some words with reasonable alternatives using a generator network. Then, they trained a discriminative network to predict if each token is an original one or a replaced one. Their experimental results showed that the representations method learned by this strategy outperforms baseline BERT model.

2.3. Semantic Search

Earlier search engines used to find documents based on lexical matches most of the time utilizing inverted indices [2]. On the contrary, semantic search techniques enable search engines improve their ranking by a better comprehension of the content of search query/question and documents/passages to be ranked, by being ale to find synonyms and/or paraphrases. The essence of semantic search is to have a fixed-sized

embedding, in other words a vectoral mapping, of all entries in a corpus of concern where the corpus can be made of sentences, paragraphs, or documents. Later at search time, a query or question is then embedded (mapped) into the same vectoral space and the most similar mappings from the corpus are retrieved. The closeness criterion can be cosine similarity or euclidean distance or any other metric of selection. Theoretically, the retrieved elements of the corpus should have higher semantic similarities with the question/query.

A taxonomy in semantic search approaches as symmetric and asymmetric semantic search. In symmetric semantic search, a query and the elements in the corpus have the same length and amount of content. Searching for similar questions can be given as an example of this sort. For symmetric tasks, queries can be flipped in training sets to provide symmetrical samples such that samples can be given as Query A-Query B and Query B-Query A. For asymmetric semantic search, a short query, question or keywords is present and the goal is to find usually a longer passage containing the anwer for the query, question, or keywords. Flipping of the query and the elements in the corpus usually does not improve the performance for asymmetric tasks. For a small corpus ($<$ 1 million elements), the similarity between the query and all elements in the corpus can be computed by cosine-similarity.

2.4. Sentence Transformers

After their release, BERT [19] and RoBERTa [22], two of the most popular transformer models used in NLP, set the state-of-the-art performances for many NLP tasks. These two models have also been used for sentence-pair regression tasks such as STS (Semantic Textual Similarity). The problem with using these models for such tasks is that both sentences should be fed into the network, resulting in redundancy in computational power consumed. Consider the case of finding the most similar sentence pair in a sentence collection of size 10,000. Since BERT uses a cross-encoder, two sentences are passed to the model and then the similarity value is computed. Such a setup would be inefficient for several pair regression tasks as this task would require approximately

50M cross-similarity computations which requires 65 hours of inference computations with BERT on a modern GPU (such as V100). Imagine trying to find the most similar question for a new question on Quora task. This would require 40 million sentence-pair inference which will require 50 hours of computing. The nature of BERT models are not suitable for semantic textual similarity tasks. Reimer and Gurevych [2] proposed a modification of the pre-trained BERT network utilizing siamese and triplet network structures to compute semantically meaningful (high semantic representational power) sentence embeddings to be used with cosine-similarity. The authors noted in their paper that this modification reduces the amount of time required for the task mentioned above from 65 hours to 5 seconds with Sentence-BERT (SBERT) while preserving the accuracy. On a GPU, SBERT is about 9% faster than InferSent and about 55% faster than Universal Sentence Encoder, therefore it is computationally a more efficient model [2]. This improvement makes BERT based models applicable to several tasks including large scale semantic similarity comparison, clustering, and information retrieval via semantic search which BERT had not been previously. The authors also evaluated their SBERT and SRoBERTa models on several common STS and transfer learning tasks and obtained state-of-the-art results. This implementation can be found in their Github repository [23].

A common strategy to tackle the problem of retrieving responses from huge corpora is mapping each sentence in the corpus to a vector space in a way that semantically similar sentence are placed close to each other. To obtain the sentence embeddings, researchers input sentences to BERT models and obtain fixed-size sentence embeddings. This can be done by averaging BERT output layer (commonly known as BERT embeddings) or by taking the output of [CLS] token [24], [25], [26]. This averaging BERT contextual embeddings methods perform even worse than averaging GloVe embeddings [2], [27].

The siamese network architecture of SBERT enables it to output fixed-sized vectors for input sentences. When these vectors are obtained, the semantic similarity between two sentences can be measured by cosine similarity or euclidean distance.

This way of calculating similarity would be much more efficient than the plain BERT method. With SBERT, the sentence embeddings calculation for 10,000 sentences takes 5 seconds and finding the most similar two would take approximately 0.01 seconds. To do this, Reimers and Gurevych [2] fine-tuned SBERT on NLI data which results in sentence embeddings that perform better than state-of-the-art methods. On seven Semantic Textual Similarity tasks, SBERT has improved the state-of-the-art performance compared to InferSent [28], compared to Universal Sentence Encoder (USE) [29], the best performing method on SentEval [30], argument similarity dataset [31] , and triplet dataset to distinguish sentences from different sections of a Wikipedia article [32].

BERT model takes two sentences separated by a special token [SEP] for sentence-pair regression task [2]. Either 12 (for base model) or 24 (for large model) of multi-head attention is applied and the output is then passed to a simple regression function to create the final label. With this pipeline, BERT set a new state-of-the-art performance for STS benchmark. This performance can even be improved using small adaptations to the pre-training process of BERT (RoBERTa [22]).

3. DATASETS

There are three different dataset used throughout this thesis work. These are namely MS MARCO, TREC CAR, and Lecture Question Answering Dataset. These three datasets are explained in the following sections 3.1, 3.2, and 3.3.

3.1. MS MARCO

MS MARCO [33], A Human Generated MAchine Reading COmprehension Dataset, is a pioneering collection of large datasets targeting deep learning and NLP researchers [4]. The very first released dataset was a question-answering dataset comprised of 100,000 real Bing questions with respective answers generated by humans. The last version released include 1,000,000 question dataset, a natural language generation dataset, a passage re-ranking dataset, a keyphrase extraction dataset, a crawling dataset, and a conversational search dataset.

MS MARCO contains 1,010,916 anonymized questions with 1,026,758 unique answers from Bing seach engine and Cortana the personal assistant of Microsoft. Every question is paired with a human generated answer. And 182,669 questions have completely human rewritten generated answers [4]. The dataset contains 8,841,823 passages selected from 3,363,535 web documents coming from Bing. For each question, a set of extracted passages obtained from Bing's search engine is provided to crowd-sourced editors to generate answers based on these passages. The editors not only generate answers for questions but also mark the passages that are relevant for a question/query. Some questions are not answerable because of insufficient or conflicting information in the questions. This collection of passages makes it possible to create natural language answers. Within the borders of this collection, a question can have one or more answers. It is also possible that a question has no answer in this candidate passage corpus.

There are three challenges based on MS MARCO dataset: the novice task, the intermediate task, and the passage re-ranking task. The first one is predicting if a question is answerable within a context of passages list, "extract and synthesize the answer as a human would" [4]. The second task, namely the intermediate task, differs from the novice task in the respect that it needs to generate a well-formed answer, meaning that the answer should make sense even without the context of the question and passages when read-aloud. The third and the last one is to "rank a set of retrieved passages given a question". This task, Passage Full Ranking, is defined as selecting top 1000 candidate passages sorted by relevance from the whole corpus of 8.8 million passages. Here, relevance of a passage to a query is defined as the likelihood of these passages containing the answer for the given query or question.

The two outstanding features of MS MARCO dataset is the size of the collections and the fact that the answers are generated by humans which gives the dataset its realistic distribution. For instance, publicly available datasets usually don't have enough size to train deep neural networks; when the size is enough the dataset is usually synthetic. However the queries in MS MARCO are real queries by real users providing a realistic distribution for real world cases. This is the reason MS MARCO has been used as a benchmark dataset in the research community for a while. The MS MARCO dataset published consists of six major components: Questions, Passages, Answers, Well-Formed Answers, Document, Question Type. A deeper insight to the dataset can be reached from its paper [4].

An example consisting of three lines from Train Triples Small set [33] from MS MARCO dataset is shown in Table 3.1. Every line in this file has one query, one relevant passage and one non-relevant passage.

Table 3.1. Samples from the MS Marco Train Triples Small [4].

Query 1: "is a little caffeine ok during pregnancy" **Relevant Passage 1:** "We don't know a lot about the effects of caffeine during pregnancy on you and your baby. So it's best to limit the amount you get each day. If you're pregnant, limit caffeine to 200 milligrams each day. This is about the amount in $1\frac{1}{2}$ 8-ounce cups of coffee or one 12-ounce cup of coffee." **Non-Relevant Passage 1:** "It is generally safe for pregnant women to eat chocolate because studies have shown to prove certain benefits of eating chocolate during pregnancy. However, pregnant women should ensure their caffeine intake is below 200 mg per day."
Query 2: "what fruit is native to australia" **Relevant Passage 2:** "Passiflora herbertiana. A rare passion fruit native to Australia. Fruits are green-skinned, white fleshed, with an unknown edible rating. Some sources list the fruit as edible, sweet and tasty, while others list the fruits as being bitter and inedible.assiflora herbertiana. A rare passion fruit native to Australia. Fruits are green-skinned, white fleshed, with an unknown edible rating. Some sources list the fruit as edible, sweet and tasty, while others list the fruits as being bitter and inedible." **Non-Relevant Passage 2:** "The kola nut is the fruit of the kola tree, a genus (Cola) of trees that are native to the tropical rainforests of Africa."
Query 3: "how large is the canadian military" **Relevant Passage 3:** "The Canadian Armed Forces. 1 The first large-scale Canadian peacekeeping mission started in Egypt on November 24, 1956. 2 There are approximately 65,000 Regular Force and 25,000 reservist members in the Canadian military. 3 In Canada, August 9 is designated as National Peacekeepers' Day." **Non-Relevant Passage 3:** "The Canadian Physician Health Institute (CPHI) is a national program created in 2012 as a collaboration between the Canadian Medical Association (CMA), the Canadian Medical Foundation (CMF) and the Provincial and Territorial Medical Associations (PTMAs)."

3.2. TREC CAR

CAR (Complex Answer Retrieval) [34] is a track that was run from 2017 to 2019 under the umbrella of TREC (Text REtrieval Conference) [35]. CAR is actually a complex of three tasks which are passage retrieval, entity retrieval, and article construction each having its own ground truth labels. The baseline approach in this thesis work has an implementation for the passage retrieval task of CAR together with the passage retrieval task of MS MARCO dataset.

Many NLP tasks including conversational answer retrieval, sub-documental retrieval, answer aggregation and many more share a common goal that is to output the answers in a formatted way such as a list ranked by relevance or size. The usual challenge here is to find the relevant answer from a single source of information. However, this approach may not be sufficient for some cases in which there are more than one source of information. In such cases, it would be necessary to blend an answer from different sources. Imagine a user searching for a specific need of information (can be assumed as a query), for example a cocktail recipe. This user would like to search for this recipe maybe in Wikipedia but not to limit himself/herself to only one source of information. He or she may want to include information from websites, brochures, books and so on. Here the blending of the answer would be from these different sources of information. One thing here to note is that as the number of the information sources increase, the likelihood of the relevant documents/passages/paragraphs being close to each other would decrease. Meaning that, the answer should be formed from text fragments that are highly distant from each other. This work historically has been made by means of manpower which is an extremely inefficient and ineffective way to solve the problem. TREC CAR challenge targets such a case and presents a dataset to be studied and researched for this scenario [15]. The challenge is to investigate different approaches to this task and to find an automated solution to retrieve relevant answers for the end user. Therefore, TREC-CAR is intended to retrieve a response from a more complex answer space and the complexity of the answer space is coming from the diversity of the resources; that is, the ultimate response to be returned to the user is

actually a combination of multiple partial answers. TREC divides this track into three sub-tracks of TREC CAR challenge: Passage Retrieval, Entity Retrieval, and Article Construction. For the scope of this thesis, only Passage Retrieval Task is relevant and only this part was explored.

TREC CAR 2017 dataset (v1.5) is formed of topics, outlines, and paragraphs obtained from an XML (Extensible Markup Language) dump (dated 20 December 2016) of English Wikipedia [15]. Each article in this dump is split into the outline of sections and the paragraphs; and then, all these paragraphs are gathered and de-duplicated. Note that, each section outline describes a complex topic and keeps the information of which paragraph belongs to which section of which article. This method provides a ground truth for the passage retrieval task. By filtering and processing, several datasets are derived from the original English Wikipedia dump. This dataset contains 29,678,367 unique paragraphs.

In TREC CAR, there are two kinds of ground truth signals are collected: automatic and manual. For each, true paragraphs are released. While the manual ground truth is assessed after participants submit runs, the automatic ground truth is derived along with the dataset from the Wikipedia dump. The automatic ground truth is released for all training sets and derived as follows: if a paragraph is in the page/section, it is considered as relevant and non-relevant otherwise. The ground truth labels is released for three different granularities: paragraph contained in section (hierarchical), paragraph contained in section hierarchy below top level section (toplevel), paragraph contained anywhere in the page (article) [15]. After resolving redirects, if a hyperlink to an entity is contained in the page/section it is defined as relevant, and non-relevant otherwise. As with paragraphs, three granularities hierarchical, top level, and article are collected [15].

The dataset is comprised of 5 folds each having approximately 580k queries. The first 4 fold (approximately 2.3M queries) is used as the training set and the last one is used as the validation set. Finally, the test set is same with TREC CAR 2017

challenge's evaluation set consists of 2,254 queries.

3.3. Lecture Question Answering Dataset

Arısoy and Ünlü gathered the MIT (Massachusetts Institute of Technology) Open Course Ware (OCW) "Signals and Systems" course videos corresponding to 11.4 hours and their reference transcriptions which contains approximately 100K words [36], [5]. The authors generated a dataset to train a Question Answering (QA) system. After dividing the reference transcripts into 1309 short passages, two annotators knowledgeable on the domain (which is signal processing for this dataset) went through these passages and generate questions for each passage. These two annotators mark the answers for the questions generated in the previous step. These answers were extracted as non-divided text fragments from the corresponding passage for each question. Since this is a dataset for a Question Answering task, the answers were selected as consecutive words from the passages. In the training set, the average word length for is 72 passages, for 11 for questions, and 24 for answers. An example passage and two question-answer pairs from the dataset are presented in Table 3.2.

Similar to the training set, an eval set has also been annotated by the authors. In total, there are 175 question-answer pairs in the test data. The average word lengths for the questions is 11 and for answers 22. The eval data contains lecture transcriptions corresponding to 1.2 hours lecture videos of Signals and Systems course prepared at MEF University. The total length of this set is approximately 10,000 words. The reference transcriptions were manually divided into 144 passages. Then, these passages were annotated using the same approach to the training set. The evaluation dataset obtained included 175 passage-question pairs for 94 passages. For the test data, the average question, answer and passage lengths have been calculated as 10, 22, and 81 words.

Table 3.2. A sample set from Lectures QA Dataset [5].

Paragraph: "now speech and images are examples of what weve referred to as continuous time signals in that they are functions of continuous variables an equally important class of signals that we will be concentrating on in the course are signals that are discrete time signals where by discrete time what we mean is that the signal is a function of an integer variable and so specifically only takes on values at integer values of the argument so here is a graphical illustration of a discrete time signal"
Question 1: "what do we mean by discrete time" **Answer 1:** "by discrete time what we mean is that the signal is a function of an integer variable and so specifically only takes on values at integer values of the argument"
Question 2: "why speech and images are examples of continuous time signals" **Answer 2:** "speech and images are examples of what weve referred to as continuous time signals in that they are functions of continuous variables"

Details of the English QA dataset are summarized in Table 3.3. Although the training set is obtained from MIT and the test set is obtained from MEF, the content and the structure of the dataset are similar.

Table 3.3. Lecture QA Dataset Statistics [5].

	TrainQA	Eval
PQ pairs	305	175
Avg P Len	72	81
Avg Q Len	10	10
Avg A Len	24	22

Compared to SQuAD1.0 [37] which is the one of the most popular datasets in the Question Answering field, this newly created QA dataset introduces longer answers containing 22 words for train and 22 words for eval dataset [36]. The passages in the dev partition of SQuAD1.0 contains 123 words and the answers for questions contain 3 words on average.

The Lecture QA dataset was originally in the form of a Question Answering task in which there are passages, questions of these passages, and answers as a subset of the corresponding passage to each question. We transformed this dataset into a passage re-ranking dataset. We labeled the passage that a question is intended for as relevant and all the other passages as non-relevant. After removing two duplicate questions, we obtained a dataset with 173 questions each having corresponding 1 relevant and 172 non-relevant passages.

4. METHODS

In this thesis, Methods chapter is divided into three sections: Baseline Implementation, Experiments on Lecture QA Dataset, and Question Answering Experiments. These three sections are explained in the following sections of this chapter.

4.1. Baseline Implementation

A typical information retrieval system is comprised of three sequential steps: Retrieval, Re-ranking, and Answer Forming. These three steps are summarized in the following as explained in Figure 4.1. The steps of this typical answer retrieval and re-ranking system that is followed by an answer generation module can be explained as follows: in the first step, a fixed number (e.g. 1000) of possibly relevant answers to a given question are retrieved from a collection of passages by an efficient approach (e.g. BM25 retrieval) [16]. In the second step, namely Passage Re-ranking, the relevancies of these retrieved passages for the given question are scored and re-ranked by a more computationally-intensive method. In the third and last step, the top-n of these re-ranked documents are used as the source for an answer generation module.

When we particularly look at the second step which is Re-ranking we have the following structure: we have a query that can be a question (e.g. "What is the population of Amsterdam") or a short expression (e.g. "the population of Amsterdam") and a list of candidate responses which can be a pre-defined answer, short paragraph, or documents list.

In a passage re-ranking system, the goal is to find the best matching passage for the given query, according to some relevance measure such that

$$p_{max} = \arg\max_{\forall p} p_r(p|q). \tag{4.1}$$

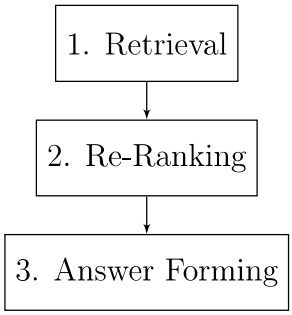

Figure 4.1. A Typical Information Retrieval System.

In the paper named Passage Re-ranking with BERT by Nogueira and Cho [16], the authors describe how they re-purpose BERT as a passage re-ranker and achieve state-of-the-art results on the MS MARCO passage re-ranking task.

The original BERT training was intended to accomplish two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). However, this model is applied to other areas of NLP such as machine translation, text classification, information retrieval, text summarization, reading comprehension, question answering, named entity recognition, natural language generation, natural language inference, or even playing chess. The authors Nogueira and Cho used BERT for another purpose which is Passage Re-ranking, utilizing BERT's representational power on written text. The source code of the implementations can be found in Github [38]. The repository contains two implementations: one for MS MARCO challenge and one for TREC CAR challenge. This implementation presents state-of-the-art results for TREC CAR challenge and is the top entry for MS MARCO passage retrieval task as of Jan 8th of 2019. It achieves 35.87 MRR@10 (on eval set) having a margin of %27 (relative) with the second best attempt for MS MARCO and also gives a MAP score of 33.5 while the best entry for TREC CAR by MacAvaney et al. [39] in 2017 achieved 14.8 MAP.

Nogueira and Cho's paper presents the implementation for a passage re-ranking step of the IR pipeline that is presented in Figure 4.1. The goal of the re-ranker is to predict a score s_i showing how relevant a candidate passage p_i is to a query q. The implementation starts from a pre-trained BERT model and fine-tunes it to the

re-ranking task using the cross-entropy loss:

$$L = -\sum_{j \in J_{pos}} \log(s_j) - \sum_{j \in J_{neg}} \log(1 - s_j).$$

Here in this formula J_{pos} represents the indices of the relevant passages and J_{neg} being the indices of non-relevant passages in most relevant 1K documents retrieved from the first stage (which is BM25 retrieval).

A coarse grained description of the algorithm is as follows:

- Feed the query as sentence A and the passage text as sentence B
- Use the BERT as a binary classifier (relevant|non-relevant)
  - use the [CLS] vector as input to a single layer neural network to obtain the probability of the passage being relevant
- Compute this probability of being relevant for each passage
- Rank the passages with respect to these probabilities (scores)

The algorithm starts with feeding the query-passage pair as Sentence A and Sentence B separated by [SEP] token and completed by [CLS] token to a BERT transformer model. Here, the query is truncated to 64 tokens at most and the passage is truncated so that query-passage pair has 512 tokens at most which is the default sequence size for BERT models. And, the BERT model is used as a binary classifier for this pair. Then, the representation on the [CLS] token for this pair is fed into a single layer neural network having the softmax probability outputs for the classification. The algorithm ends with re-ranking the passages by the softmax probability of being relevant of each passage for a given query. The output of this binary classification is the classes Relevant or Non-relevant.

Even though in an unsupervised way, the official pre-trained BERT models have seen the Wikipedia corpus [40]. To prevent this leak of test data in to training data, the BERT re-ranker is pre-trained only on the half of Wikipedia corpus that is included

in TREC-CAR's training set. For the fine-tuning of this model on the other hand, query-passage pairs are generated by retrieving the top-10 passages from the entire TREC-CAR corpus with BM25 technique. At the end of this retrieval, 3M queries each having 10 passages, there are 30M example query-passage pairs are obtained for fine-tuning. The model is fined tuned for 400k iterations, equivalently 400k iterations each having 32 query-passage pairs, there are 12.8M training examles. This 12.8M training example set corresponds to only 40% of the whole training set. Fine-tuning the model further does not increase the performance on the dev set [16].

4.2. Experiments on Lecture QA Dataset

As the main focus of this thesis, it is aimed to formulate a similar pipeline for a question answering system with retrieve and re-rank answers approach. This approach has been applied to Lecture Question Answering Dataset, with a modern PyTorch library and transformer models in Huggingface repository.

Although the original dataset is actually generated for a typical question answering task of NLP [5]; in our work, this dataset is re-purposed for a passage re-ranking task. In the original dataset, there were passages and question-answer pairs for each passage. We extracted each question and considered every passage as a candidate passage containing the answer. It is assumed that there is only one relevant passage for each question in the dataset since every question has an answer in its corresponding passage. The ultimate goal is to retrieve the passage containing the answer. And this is achieved by retrieving a small number of candidate passages from the corpus via TF-IDF (Term Frequency Inverse Document Frequency) approach and then re-rank this short-list using the relevance scores obtained via Cross-Encoders.

As explained in this chapter, a typical question answering pipeline consists of three phases which are retrieval and re-ranking, and answer generation. In our implementation, the retrieval phase is done via a computationally less expensive method such as TF-IDF. And then, it is followed by a re-ranking step which utilizes a more compu-

tationally expensive machine learning method such as BERT. This generic framework is also followed in the work in this work.

4.2.1. TF-IDF Retrieval Phase

In this thesis, a TF-IDF retrieval system is implemented from scratch, meaning that, no specific library in Python is used for TF-IDF.

4.2.1.1. Preprocessing. The very first step Lower Case Conversion is a simple step where all the characters are converted to their lower-case forms.

In Punctuation Removal step, a lookup for a symbol list is carried on. When one of these characters is met in the text, it is replaced with a space character. Multiple spaces are reduced down to a single space character. And, lastly commas are removed from the text. The removed punctuation marks can be seen in Table 4.1.

Table 4.1. Removed Punctuation Marks and Symbols.

!	”	#	$
%	&	(	)
*	+	-	.
/	:	;	<
=	>	?	@
[	\	]	^
_	′	{	|
}	~	\n	

Apostrophe Removal step is as easy as just removing the apostrophes from the text.

In Stop Word Removal phase, the text is tokenized with the tokenizer of the NLTK library in Python. Then, whenever a stop word is come across, it is removed

from the text. The stop word list is also obtained from the NLTK library in Python.

Text Number Conversion step is comprised of just changing a token if its an integer with its string version.

For Stemming, PorterStemmer from NLTK libary of Python is used. And, Stemming is applied to words when possible.

The steps in preprocesing of the text is summarized in Figure 4.2. It can be seen that some of the pre-processing steps are applied repeatedly. The purpose of this is cleaning the data further since the output of the stemming step may contain punctuation marks, text numbers, and stop words.

4.2.1.2. Calculation of TF-IDFs. After preprocessing of the text, document frequencies for each word are calculated. Here, the document frequency refers to simply the number of different passages that a word appears in. Document frequencies (DF) are converted to inverse document frequencies (IDF) by simply dividing the collection size by the number of the passages that the words appears in and taking the logarithm of the result. The formula for calculation of IDF values is

$$\text{IDF} = \log(\frac{\text{Collection Size} + 1}{\text{df} + 1}).$$

After calculation of inverse document frequencies for each word in the corpus, term frequency (TF) value for each word against each passage is calculated. Term Frequency is assumed to be equal to the number of times a term appears in a passage divided by the total number of the words in that passage.

TF-IDF value of a word in a passage is the multiplication of the TF of that word with the IDF value of that word.

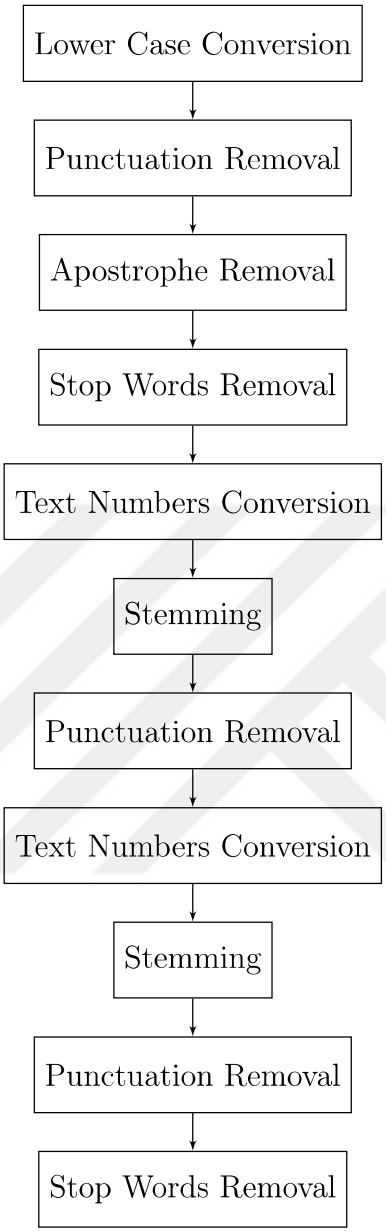

Figure 4.2. Preprocessing Steps.

4.2.1.3. Retrieve Docs for Each Question. After obtaining the TF-IDF values of words, the passages are vectorized with these calculated TF-IDF values. Each question is also pre-processed with the same preprocessing pipeline and vectorized with the TF-IDF vectorizer obtained in the previous step.

The similarity of a question with a passage is calculated using the cosine similarity of the TF-IDF vectors of the query and the passage. For each question, the cosine similarity of the the question is calculated with each passage in the corpus. The

passages are then ranked with respect to this similarity metric. Lastly, the first 5 passages with the highest similarity to the given question are retrieved from the whole corpus as the result of the retrieval step.

This process is repeated for each question. At the end, top 5 most similar passages to a given question are retrieved from the corpus. Despite the number of steps applied during the retrieval phase, this phase is computationally not that expensive. Therefore, the retrieval step is used to short-list candidate passages list for a given question.

These retrieved candidate passages are then passed to the re-ranker step which is the second step in the IR pipeline of this thesis.

4.2.2. Re-ranking

4.2.2.1. SentenceTransformers. SentenceTransformers is a Python Framework to be used for text, sentence, and image embeddings [41]. It is based on the research work of Reimers and Gurevych [2]. These models are used to create embeddings for sentences. In some of the pre-trained multilingual models, the sentence embeddings are available in more than 100 different languages. These embeddings lie in a vector space in which cosine similarities between these embeddings can be used to quantize the similarities among the sentences. Moreover, other similarity metrics (e.g. euclidean similarity) can also be used to measure similarity. This funtionality is useful for NLP tasks such as paraphrase mining, semantic search, semantic textual similarity, and many others. SentenceTransformers is developed with PyTorch [42] and Transformers [10] which creates its power to have a huge collection of pre-trained models that can be fine-tuned for down-stream tasks. There are even already fine-tuned (released) models in SentenceTransformer's repository.

SentenceTransformers framework not only releases Bi-Encoder which can be used to create sentence embeddings but also Cross-Encoder models that take two input sentences (such as a query, question, passage, or short document) and quantize the simi-

larity between them. This process is done without computing the vector embeddings for each sentence independently. The pre-trained models released by the owners of the paper are fine-tuned on MS MARCO passage retrieval task dataset.

There are two types of networks in SentenceTransformers approach: Bi-Encoder and Cross-Encoder. Bi-Encoders produce individual embeddings for individual sentences. Two different sentences are fed into a siamese transformer structure to obtain two different embeddings u and v. Later these two embeddings (u and v) are to be compared with cosine similarity or a softmax classification layer. On the other side, two input sentences are fed into Cross-Encoder architectures simultaneously. The transformer network is followed by a classifier to obtain a score between 0 and 1 measuring the relevance of the passage to the query. During this process, a Cross-Encoder does not create embeddings for each input sentences individually. Also, it is not possible to input individual sentences into a Cross-Encoder. Cross-Encoders outperform Bi-Encoders for passage re-ranking tasks [2]; however, they lack the embeddings for individual input sentences which can be useful for several NLP tasks.

Cross-Encoders are to be used when a pre-defined set of text pairs are to be relevance scored. Bi-Encoders are used when individual vectoral representations for each input sentence is necessary. For instance, clustering 10k sentences using Cross-Encoders would require to get a relevance score for approximately 50M sentence-pair combinations. But; with a Bi-Encoder, the embeddings for each sentence can be computed individually in a short period of time. Then, it is fast to compare the fixed-sized embeddings in a vector space.

For Retrieval and Re-ranking task, the original SentenceTransformer approach uses the same pipeline outlined in Figure 4.1. with one difference from the baseline implementation that is SentenceTransformer's Retrieval step uses Bi-Encoders instead of TF-IDF. One point to mention here is that the Reimers and Gurevych consider a corpus of 100 passages as a long list but for production systems the usual case is having thousands, millions, or even billions of passages in the information corpus [2]. Although

this retrieval method is a semantic search method meaning that it can recognize synonyms, acronyms, and spelling variations, it is computationally more expensive than lexical methods. With a lexical approach such as TF-IDF, it is much faster to reduce down the size of the candidate passages for a given query. Since the retrieval phase has to be efficient for huge passage collections, TF-IDF method has been implemented in this thesis. After the retrieval phase, the re-ranker improves the output of the retrieval phase utilizing a more computationally expensive method. In Cross-Encoders, a query and a passage is passed to the transformer model simultaneously. The end product of the transformer network is a score between 0 and 1 that indicates how relevant the query/passage tuple to each other. The major advantage of using Cross-Encoders is the higher performance since Cross-Encoders perform the attention operation on the query and the passage at the same time (cross attention). Pre-trained Bi-Encoders for the retrieval phase are also shared through SentenceTransformer library.

Since each task in NLP has its own characteristics and challenges, it makes sense to produce embeddings depending on the task to be achieved. Thus, training techniques and the architectures to be used depend on the task of concern. The simplest network architecture mapping a variable size text input to a fixed-size dense vector is given in Figure 4.3. In this structure, the input text is fed into a transformer network and then contextualized embedding for each token in this sentence is obtained. After this point, a pooling [43] layer is needed to produce a fixed-sized vectoral representation of the whole text input, the vectoral representation is named as vector u in Figure 4.3. There are many options available for the pooling approach while the simplest one being mean-pooling. What mean-pooling does is averaging all the contextualized embeddings for each token passing in the input text. The end result of this algorithm is a fixed-sized vector regardless of the size of the input text. The size of this so-called fixed-sized vector depends on the size of transformer used such that it would be a vector of size 768 for base size models such as BERT-Base or a vector of size 1024 for large size models such as RoBERTa-Large. In SentenceTransformer models, this fixed-sized embedding output can also be downsized with a fully connected dense layer with Tanh activation.

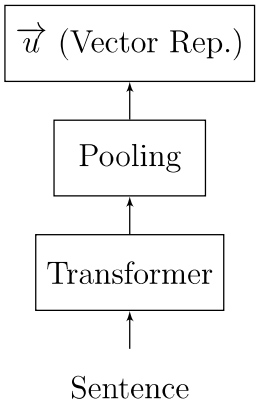

Figure 4.3. Basic Training Architecture [1].

The loss function also plays an important role in the fine-tuning of a pre-trained Huggingface model. The logic for selecting the best loss function for the training depends on the task in hand. For the pre-trained model to be fine-tuned to the downstream task, it is necessary to indicate which text-pairs are similar to each other and which are not so that the network can learn a mapping of these texts to a vector space where similar texts are mapped closer to each other than the dissimilar ones. In SentenceTransformers implementation, the simplest way of scoring sentence similarity is employed. A similarity/relevance score between 0 and 1 is given for each sentence pair. In our trainings, we selected 0.9 for text pairs that are relevant to each other and 0.1 for the pairs that are not relevant to each other.

In the softmax classifier objective function case, the two input sentences are fed into the transformers, fixed-size output embeddings are obtained via pooling and the two fixed-sized vectors obtained for each sentence are then concatenated as (u, v, and |u-v|) to form one, long representation. This vector is then passed to the softmax classifier which will compute the probabilities for three different classes for NLI (Natural Language Inference) task which are entailment, neutral, and contradiction.

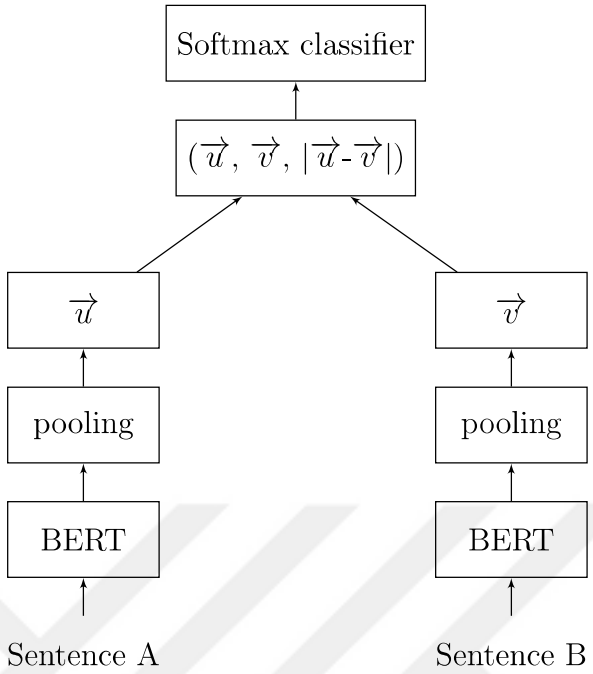

Figure 4.4. Bi-Encoder architecture with softmax classifier objective function [2].

In Figures 4.4 and 4.5, two different approaches for calculating the similarity between sentences are shown. In Figure 4.4, a softmax classifier is applied to the outputs of the siamese transformers for each text in the text pair. In Figure 4.5, the similarity between the two input sentences is calculated via cosine-similarity between the fixed-sized vectoral representations. The two transformer networks have tied weights in these architectures; this approach is called siamese network structure.

There are other features of SentenceTransformer implementations such as knowledge distillation, data augmentation strategies, unsupervised training of sentence embeddings; but, these features are not needed in our experiments as we have enough training data coming from MS Marco sets and could manage to train our models at least up to the size of base models (e.g. BERT-Base, RoBERTa-Base).

One strong side of SentenceTransformers is that it is initialized with pre-trained transformer models. This also reduces down the time necessary to fine-tune SentenceTransformer models.

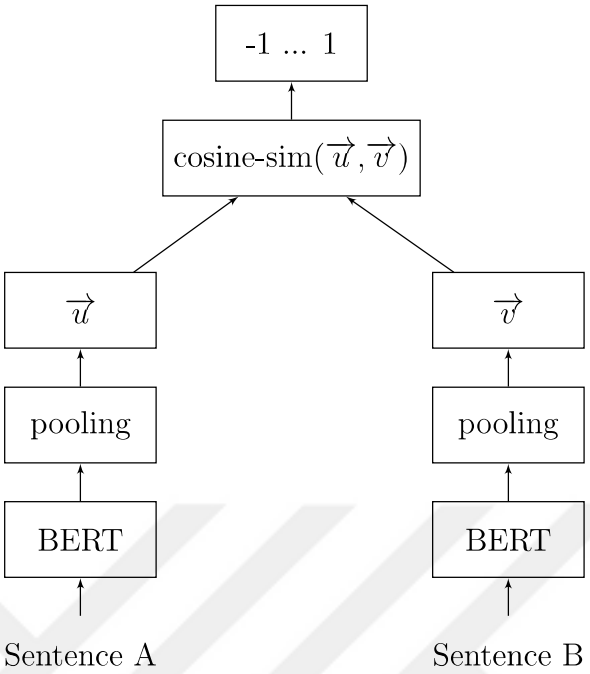

Figure 4.5. Bi-Encoder architecture with cosine similarity objective function [2].

SentenceTransformers have a pooling operation to the outputs of transformers to obtain a fixed-sized sentence embedding. The three different pooling strategies are using the output of the CLS-token, computing the mean of all output vectors (MEAN-strategy), and computing a max-over-time of the output vectors (MAX-strategy) [2] while the default configuration being MEAN-strategy.

Schroff et al. create siamese and triplet networks to update the weights of BERT/RoBERTa model while fine-tuning [44]. The network structure depends on the training data. The authors experiment with Classification Objective Function, Regression Objective Function, and Triplet Objective function.

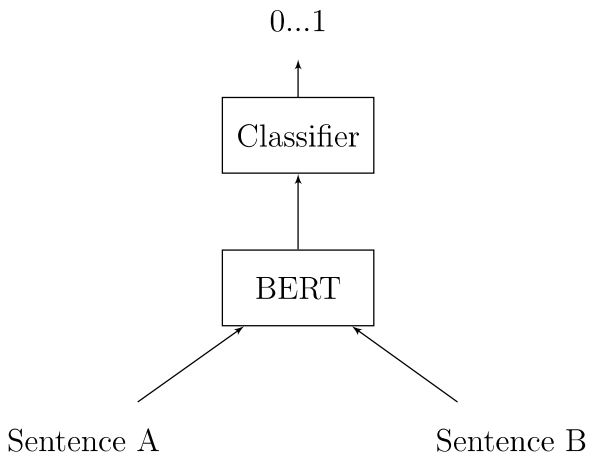

Figure 4.6. Cross-Encoder Architecture [3].

The authors train SentenceTransformer architectures on a dataset combination of SNLI [45] and Multi-Genre NLI dataset [46]. The SNLI dataset has 570,000 sentence pairs with one of contradiction, entailment, and neutral labels. MultiNLI dataset, on the other side, has 430,000 pairs of sentences. Fine-tuning of SentenceTransformers is done with a 3-way softmax-classifier objective function for one epoch. A batch size of 16 is used with Adam optimizer and the learning rate is set to 2e-5 with a linear learning rate warm-up over 10% of the training data. The default strategy for pooling is selected as MEAN. The architecture for Cross-Encoders can be investigated in Figure 4.6.

4.2.2.2. Pre-trained Cross-Encoders Models As Rerankers. After short-listing the candidate answers for a given question, the second step is re-ranking these candidate answers with a more computationally expensive but a more intelligent solution to find the most relevant passage for a given question.

In the baseline approach summarized in Figure 4.1, a BERT model is utilized as a re-ranker. A question and a passage is fed to the model, the BERT model is used obtain representations. The [CLS] vector (classification) is used as input to a single layer classifier neural network to obtain the probability of the passage being relevant [16]. The softmax output layer of this classifier network is used to calculate

the probability of a passage being relevant to a question. And then this probability is interpreted as the relevance score of that passage against the question.

In this thesis, this re-ranking process is replaced with the Cross-Encoders of SentenceTransformer. Instead of using plain BERT models as classifiers, a more sophisticated approach to the re-ranking process is applied through Cross-Encoders.

In SentenceTransformer library, there are shared pre-trained and fine-tuned Cross-Encoder models. These models are fine-tuned on MS Marco Passage Retrieval dataset which is explained in Section 3.1. These models [41] can be used for semantic search; in other words, for given keywords or a search phrase or a question, the model can find text passages relevant for this searched item. The size of the training data these models are fine-tuned on includes approximately 500k examples of MS MARCO dataset.

In the re-ranking phase, the pretrained and shared cross-encoder model 'cross-encoder/ms-marco-MiniLM-L-12-v2' is used. This model outperforms the other pre-trained/fine-tuned models in the released pre-trained/fine-tuned model repository as can be seen in Table 4.2. The maximum token length for this models is set to 512 tokens meaning that any question or passage longer than this length will be trimmed after this maximum length. The cross-encoder model is used as a relevance score generator in this scenario. Note that the short-listed candidate responses to a given question are already obtained in the previous retrieval phase. These short-listed candidate passages are then paired with the question of subject.

Table 4.2. Released Fine-tuned Model Performances on MS MARCO, runtimes were computed on a V100 GPU using Huggingface Transformers v4 [2].

	MRR@10 (MS Marco Dev)	**Docs/Sec**
BERT-Large as Re-ranker(Baseline)	36.53	-
ms-marco-TinyBERT-L-2	30.15	9000
ms-marco-TinyBERT-L-4	34.50	2900
ms-marco-TinyBERT-L-6	36.13	680
ms-marco-electra-base	36.41	340
ms-marco-TinyBERT-L-2-v2	32.56	9000
ms-marco-MiniLM-L-2-v2	34.85	4100
ms-marco-MiniLM-L-4-v2	37.70	2500
ms-marco-MiniLM-L-6-v2	39.01	1800
ms-marco-MiniLM-L-12-v2	**39.02**	960
pt-tinybert-msmarco	28.80	2900
pt-bert-base-uncased-msmarco	34.75	340
pt-bert-large-msmarco	36.48	100
electra-base-msmarco	36.89	340
bert-multlngl-pssge-reranking-msmarco	35.54	330
distilbert-cat-margin_mse-T2-msmarco	37.88	720

The relevance score for each question-passage pair is obtained using the Cross-Encoder models. Cross-Encoder models output a relevance score between -14 and +14. This relevance score is then normalized to a value between 0 and 1. Normalized relevance scores are calculated as

$$\text{Normalized Relevance Score} = \frac{\text{Relevance Score} + 14}{28}.$$

This normalization is only for the sake of more meaningful relevance scores. It doesn't affect the ranking or any other process. The candidate passages list is then re-ranked

according to these relevance scores obtained. The expected result of this second step, namely re-ranking step, is a better ranking of the passages in the short-listed passages list.

4.2.2.3. Custom Fine-tuning with Huggingface. We also used pre-trained transformer models in Huggingface repository to fine-tune it in Cross-Encoder structure. We use MS MARCO dataset for this fine-tuning because the pre-trained and fine-tuned models cross encoders are fine-tuned with this dataset. The authors state that they have fine-tuned these cross-encoder models with approximately 500k training samples from MS Marco dataset and give no further information about the fine-tuning dataset.

We have used Train Triples Small set [33]. This set is a 7.9 GB tar.gz. extension file. When extracted, it turns into a 29GB size TSV (Tab-Separated Values) file. In each line, there is a query and one relevant and one non-relevant passage for this query. This is why the relevant and non-relevant class sizes are always equal to each other as the number of the lines used increases.

For training batch size is selected 32 for distilled models (distilroberta-base, distilbert-base) and 16 for the other models. The reason for this is that the size of the models does not allow for training in RTX2080 Ti GPU resources that are available for us. The number of labels for the models is selected as 1 resulting in Cross-Encoders to be regression models that output a continuous score between 0 and 1. The max_length is not set for Cross-Encoder models; thus, the maximum length for input sequences is set by the models' default coming from Huggingface models. Longer sequences (in terms of tokens) are to be truncated. Also, the default activation function which is sigmoid function is used during the trainings.

First 150,000 lines of the Train Triples Small file is selected. The first 125,000 lines are assigned as training set. Then, the first 15,000 lines of the remaining part are selected as the dev set and the last 10,000 lines are marked as test set. For every line in MS MARCO set, there are two samples comprised of one relevant one non-

relevant passage. During the forming of training, dev, and test sets, every query-passage pair is fed as query-passage and passage-query. Therefore, for each line in MS MARCO dataset, we obtain 4 training/dev/test samples. While training the models, the relevant query-passage pairs are labeled as 0.9 and the non-relevant query-passage pairs are labeled as 0.1. The decimal scores come from the regression nature of the Cross-Encoder models. Number of epochs are selected as 1 or 4 depending on the specific experiment. The number of warmup steps is calculated as the following formula which corresponds to 10% of the training data. Number of warmup steps is calculated as

$$\text{Warmup Steps} = (\text{Training Data Size}) * (\text{Number Of Epochs}) * 0.1.$$

4.2.2.4. Further Fine-tuning with Lecture QA Dataset. In this part, we continue fine-tuning of our best performing model in the previous section. This process is called as further fine-tuning in this thesis. For the further fine-tuning dataset, the training dataset of Lecture QA Dataset prepared by Arısoy and Ünlü [5] was used. At first, questions and passages containing the answers for these questions have been marked as relevant question-passage pairs and then the rest of the passage collection have been marked as non-relevant question-passage pairs. Using this approach, we obtain 304 relevant question-passage pairs and 304 * 303 = 92,112 non-relevant question-passage pairs. There is a huge class imbalance in this dataset. We multiply the relevant pairs by 304 times and use it in the training.

Additionally, we employ another approach to solve the class imbalance problem. Since we have 1 relevant and 304 non-relevant passages for every question, we take this relevant passage and we randomly take 35 of all non-relevant passages. The class imbalance problem still exists but it is less severe. This will be the dataset for second further fine-tuning experiment and the model using this second further fine-tuning dataset is called further-fine-tuning-modified.

4.3. Question Answering Experiments

Beyond for a passage re-ranking task, Cross-Encoder models can also be used for a question answering task. Here, a question answering task refers to the Natural Language Processing task which is basicly finding the answer of a question as a consecutive subset of a passage. In this work, we re-purposed Cross-Encoder models so that they can also be used for such a task. The idea stems from the fact that we can obtain a high quality relevance score between two sentences. Given a question and a passage in which the answer of the question lies as a consecutive subset of the corresponding passage, Cross-Encoders are used to extract the answer for the question. Each question is paired with all the possible word spans of the passage on the other side. The relevance score for each pair is calculated. Then, the most relevant word span is returned as the answer of the question. Note that this is a computationally expensive process.

5. EVALUATION

The evaluations of the approaches in this thesis are divided into three parts: Baseline Experiments, Re-ranking, and Question Answering.

5.1. Baseline Experiments

As presented in Table 6.1, the evaluation results for the experiments on the baseline approach are given in MAP (Mean Average Precision) and MRR (Mean Reciprocal Rank).

5.1.1. MAP

AP (Average Precision) is calculated as follows:

$$AP = \sum_{k=1}^{n} p(k)rel(k).$$

After then, MAP score is calculated as given in the following formula:

$$MAP = \frac{1}{|U_{All}|} \sum_{u=1}^{|U_{All}|} AP(u).$$

5.1.2. MRR

MRR (Mean Reciprocal Rank) score is calculated as the following:

$$MRR = \frac{1}{Q} \sum_{i=1}^{Q} \frac{1}{rank_i}.$$

5.2. Re-ranking

There are four different metrics shared for our re-ranking experiments: mean NDCG, MAP, MRR, and catch@n. The computation methods for these metrics are presented in Sections from 5.2.1 to 5.2.3.

5.2.1. NDCG

NDCG (Normalized Discounted Cumulative Gain) calculation is given in the following formulas. First CG (Cumulative Gain) is defined as follows for a ranking result by the system where rel_i represents the relevance score of a document,

$$CG = \sum_{i=1}^{N} rel_i.$$

To discriminate between front and end of a ranking, DCG (Discounted Cumulative Gain) is calculated as follows:

$$DCG = \sum_{i=1}^{N} \frac{rel_i}{\log_2(i+1)}.$$

When DCG is divided by iDCG which stands for ideal Discounted Cumulative Gain that can be attained by a system, NDCG is obtained as in the following formula,

$$NDCG = \frac{DCG}{iDCG}.$$

5.2.2. MAP and MRR in Re-ranking Experiments

One thing to note here is that although we have implemented both MAP and MRR metrics, we have shared only MRR scores. The reason for this is that MAP and MRR evaluations are equal to each other for only-one relevant passage re-rankings.

If the labelings for our system was other than binary or if there were more than one relevant passages that can be retrieved and re-ranked, these two metrics would be different and worth to be shared separately. Since this is not the case for the scope of this thesis, we have only shared MRR scores for our evaluations but not MAP which has the same numeric value with MRR.

5.2.3. Catch @n

catch@n is a simple metric that denotes the number of questions (out of all 173) that the re-ranking method catches the relevant passage at n'th document. The maximum value for n is 5 because of the fact that only 5 documents are retrieved using TF-IDF approach. The increasing number towards catch@1 emphasizes a better ranking performance by the system.

5.3. Question Answering

For the question answering experiments three types of evaluation metrics are shared: the official SQuAD eval script's F1 score, Rouge, and BLEU.

5.3.1. SQuAD Official Evaluation Script

In this evaluation metric, SQuAD's official evaluation script [47] has been used. This script outputs an F1 score. This score is compared with the paper from Arısoy and Ünlü [5].

5.3.2. Rouge

Rouge is a common metric usually used to evaluate text summarization, machine translation, and question answering systems. Rouge is a primitive but strong metric that evaluates the overlap between hypotheses created by the system and ground truth references. The evaluated metrics in this work are Precision, Recall, and F1 metrics at

Rouge-1, Rouge-2, and Rouge-l levels. Although the results for Precision and Recall results have been computed for different models, these values are not shared in Results chapter. Instead F1 scores for unigrams, bigrams, and Longest Common Subsequence levels are shared for the best performing model.

5.3.3. BLEU

Bilingual Evaluation Understudy (BLEU) is an evaluation metric to evaluate the quality of hypothesis texts produced by systems with respect to golden standard reference texts. Here, the texts that are evaluated are generally machine translation or summarization outputs and references. The goal is to evaluate the correspondence between the system output and reference sentences. BLEU has traditionally been used for translation systems and one of the earlier evaluation frameworks that is proposed to have a high correlation with manual evaluations.

BLEU always outputs a value in the range between 0 and 1, similar to Rouge. In this thesis, BLEU scores are shared in different levels at 1, 2, 3, and 4.

6. RESULTS AND DISCUSSION

In this chapter the results obtained in the experiments are presented and discussed. Similar to Methods chapter, this chapter is also divided into three sections: Baseline Approach, Re-ranking Results, and Question Answering Results.

6.1. Baseline Approach

Our experiments' on the baseline approach results are shown in Table 6.1. As can be seen in this table, there are 4 different parameters of our experiments: query size, passage size, BERT model that training is initialized with, and batch size. We make changes to these hyperparameters one by one and observe their effects of the performance of the model. The evaluation of the performances were measured with MAP and MRR scores. We will be making comparisons only based on MAP scores which are aligned with MRR.

The authors Nogueira and Cho [16] give two baseline results which are presented in rows 1 and 2 in Table 6.1. The first baseline uses BERT Large model while the second uses BERT Base model; nevertheless, these two models are not pre-trained models officially released by Google. The pre-training of these two models are done separately by the authors. We also re-produced these baselines in Google's Colaboratory platform [48]. The MAP scores reported by the authors are 0.336 for BERT-Large and 0.310 for BERT-Base. First, we start by trying the same experiment with the same configuration without changing anything, this experiment is presented in the third row. Although, the authors report 0.336 MAP score on their experiments for the large model, we measure this performance as 0.333. In rows from 4 to 7, we measure the effect of batch size on the model performance. As can be seen, we did not observe any major decrease until batch size of 4. Batch size of 2 loses approximately 0.03 map score; thus, we continued our experiments with the batch size of 4. Then, we decreased the upper limit for the size of query from 64 to 32 and did not observe a major difference

in the performance. We also tried to reduce the total sequence length of the BERT model from 512 to 256. This manipulation decreased MAP score from 0.331 to 0.316. Finally, we initialized the pre-trained BERT model from the officially released BERT-Base model instead of starting from a custom pre-trained model. This change increase the performance of BERT-Base model from 0.310 to 0.320. The TREC CAR dataset is composed of Wikipedia articles, header, sections and paragraphs. Since the officially shared BERT models are pre-trained on Wikipedia corpus, the authors avoid using the shared models. We proved that this concern has a solid ground. When the pre-trained BERT model has seen the Wikipedia corpus during the pre-training, this dev and test data leak into training data boosts the performance of the model to some degree. Although this test data leak into training data happens in an in-direct way, it has an effect on the model performance.

Table 6.1. Baseline Experiments on TREC CAR Dataset.

Exp. Id	**Query**	**Passage**	**BERT Model**	**Batch**	**MAP**	**MRR**
1	64	512-q	Large	32	**0.336**	**0.479**
2	64	512-q	Base	32	0.310	-
3	64	512-q	Large	32	0.333	0.477
4	64	512-q	Large	16	0.331	0.475
5	64	512-q	Large	8	0.331	0.475
6	64	512-q	Large	4	0.331	0.475
7	64	512-q	Large	2	0.296	0.431
8	32	512-q	Large	4	0.331	0.475
9	64	256-q	Large	4	0.316	0.456
10	64	512-q	Official-Base	4	0.320	0.454

After the first series of experiments, the experiment 10 in Table 6.1 has been selected as a baseline for a second series of experiments. In this second series of experiments, there are two hyperparameters of concern: hidden size and number of layers of the BERT model that the trainings initialized with. In their repository [40], Google shares BERT checkpoints that are pre-trained on Wikipedia corpus with varying

hidden size and number of layers. The varying hidden sizes are 128, 256, 512, and 768. On the other side, the varying number of layers are 2, 4, 6, 8, 10, and 12. In this cartesian variation of the released models; BERT-Tiny corresponds to 2-Layers of hidden size 128, BERT-Mini corresponds to 4-Layers of hidden size 256, BERT-Small is 4-Layers of hidden size 512, BERT-Medium is 8-Layers of hidden size 512, and BERT-Base is 12-Layers of hidden size 768. For the sake of completeness, BERT-Large has 24-Layers of hidden size 1024. We have initialized the fine-tuning phase with each of these pre-trained models and measured the performance in MAP score. The results of this second experiment series can be seen in Table 6.2. The configuration in experiment 10 in Table 6.1 is followed for all these 24 experiments. To put into the words; upper limit of 64 tokens is used for the query, the maximum sequence length of 512 is used and the size of the doc is adaptive depending on the size of the query, the batch size is 4, 40k steps of warmup steps used to set the learning rate, 400k steps of training is applied to the officially release pre-trained BERT model checkpoints.

Table 6.2. Model Size Experiments on TREC CAR Dataset.

	H-128	**H-256**	**H-512**	**H-768**
L-2	0.093	0.149	0.184	0.213
L-4	0.098	0.196	0.252	0.264
L-6	0.175	0.207	0.269	0.285
L-8	0.167	0.238	0.282	0.306
L-10	0.171	0.238	0.282	0.314
L-12	0.153	0.245	0.291	**0.320**

According to Table 6.2; we observe that larger models, with larger hidden sizes and more layers contained in the transformer structure, perform better than smaller models. Another observation is the effect of decreasing hidden size has a bigger negative impact than reducing the number of layers in the transformer model.

6.2. Re-ranking Results

The evaluation results shared by Reimers and Gurevych for pre-trained Cross-Encoder models were given in Table 4.2. Additional to the pre-trained Cross-Encoder model performances, the evaluation of the baseline approach was added to this table in the first row named "BERT-Large as Re-ranker (Baseline)". In the second section, indicated by horizontal lines, the pre-trained models that are released through the SentenceTransformer library have been evaluated. In the third and last section, the performance evaluations for the passage re-ranking transformer models that are already released through Huggingface have been presented. The performance evaluations are made with MRR@10 metric on MS MARCO dev set. As can be seen from the this table, the best performing model is the model named as "ms-marco-MiniLM-L12-v2". Therefore, this model has been evaluated on Lecture Passage Re-ranking Dataset.

In Table 6.3, a comparative evaluation on Lecture Passage Re-ranking Dataset can be seen. There are two evaluation metrics in this table: Mean-NDCG and MAP. Even though MRR metric evaluation script has also been implemented in our work, MRR results has not been shared in this report. The reason for this is that MRR and MAP calculations give same numeric result for re-ranking tasks where there is only one positive, or relevant, candidate. The evaluation of TF-IDF retrieval step is placed at the top of this table. After TF-IDF, the best performing pre-trained Cross-Encoder model released with SentenceTransformer library, ms-marco-MiniLM-L12-V2 has been used to re-rank the top-5 passages retrieved with TF-IDF method and the evaluation results are shared. In the third region, indicated via horizontal lines, our fine-tuned transformer models that are initialized by the pre-trained models with their corresponding names have been shared.

Table 6.3. Re-ranking Results Mean-NDCG and MAP Evaluations.

	Model	Mean-NDCG	MAP	Size(MB)
1	**TF-IDF**	0.771	0.712	NA
2	**ms-marco-MiniLM-L-12-v2**	0.839	0.801	127
3	**bert-base-multlngl-cased**	0.827	0.786	681
4	**bert-base-multlnl-uncased**	0.829	0.787	641
5	**distilbert-base-uncased**	0.815	0.770	256
6	**distilroberta-base**	0.816	0.771	316
7	**dstlbert-base-uncased(4 ep.)**	0.775	0.717	256
8	**distilroberta-base (4 ep.)**	0.800	0.750	316
9	**electra-small-discriminator**	0.826	0.785	52
10	**albert-base-v1**	0.837	0.800	45
11	**albert-base-v2**	0.832	0.793	45
12	**further-fine-tuning**	0.838	0.800	45
13	**further-fine-tuning-modified**	**0.853**	**0.821**	**45**

In Table 6.3, it can be seen that when the re-ranking step is applied, the rankings obtained in the retrieval phase is always improved. When the best performing pre-trained and fine-tuned on MS MARCO dataset model shared by SentenceTransformer library is applied at the re-ranking step, we obtain an increase from 0.771 to 0.839 in Mean-NDCG and an increase from 0.712 to 0.801 in MAP score.

Instead of using an already pre-trained and fine-tuned model, we also fine-tuned our own re-ranker models. When the multilingual versions of BERT-Base size models are used, we achieve approximately 0.83 in Mean-NDCG and 0.786 in MAP score. When we use distilled models such as distilbert-base-uncased or distilroberta-base, we observe comparative results with slightly less scores in Mean-NDCG and MAP. We achieve 0.815 Mean-NDCG and 0.770 in MAP for distilbert-base-uncased; and 0.816 Mean-NDCG and 0.771 in MAP for distilroberta-base. These two distilled models are smaller than half size of multilingual BERT-Base models; but, they perform almost

at the same level confirming the study [21]. After this point, we tried continuing fine-tuning of these distilled models to see if we can see any improvement in the performance. As is clear in Table 6.3, fine-tuning for 3 more epochs instead of stopping at 1 epoch didn't result in any improvement in the performance; instead, decreased the performance.

Beyond what has been achieved with distilled models in the previous step, we also tried google/electra-small-discriminator and achieved same performance with Base size models with an even smaller model.

We also tried albert-base-v1 and albert-base-v2 models. These two models are the two smallest models that we have experimented with. In contrast, these two models (especially albert-base-v1) outperformed other models including largest models bert-base-multilingual-cased and bert-base-multilingual-uncased. Consequently, we also achieved higher performances with smaller models in this thesis proving our point.

Since the best performing custom fine-tuned model is albert-base-v1, we continue with further fine-tuning of this model. Starting from already fine-tuned model, we further fine-tune this model as explained in Section 4.2.2.4. Although we observe a slight increase in Mean-NDCG and no change in MAP score, this further fine-tuning decreased the model's performance on catch@1 evaluation. We applied a randomized downsizing to the non-relevant question-passage pairs as explained in Section 4.2.2.4. By this method, we achieved a Mean-NDCG score of 0.853 and MAP score of 0.821 which stands for the best performing model in the thesis.

In Table 6.4, we give a finer grained and a more detailed results of the performances of the model pool that are already shared in Table 6.3. In this new table, evaluation metrics are catches@1, catches@2, catches@3, catches@4, and catches@5. Here, catch@n corresponds to the ratio of questions for which the relevant passage is ranked in the first n re-ranked results. For instance, TF-IDF retrieval step catches the relevant passage in the first rank for 58.5% of all questions. In a similar manner,

this retrieval step catches relevant passage in the first two ranking for 80.5% of all questions. It can be expected that the most important evaluation metric in this table is catch@1 metric. Thus, catch@1 has been chosen as the determining metric for the comparisons of transformer models. The analysis in this table is made to understand the performances of the models in top ranks such as the first and the first two ranks. We observe that the best performing model for the evaluation metric catch@1, catch@2, and catch@3 is further fine-tuned albert-base-v1 model, named further-fine-tuning-modified (row 13). For the metric catch@4, we see ms-marco-MiniLM-L-12-v2 and bert-base-multilingual-uncased perform equal to further fine-tuned albert-base-v1 with the performance of ranking of the relevant passage at top 4 places in all of the questions out of 164 questions that retrieval phase retrieved the relevand document in the first 5 rankings. The most important observation of this thesis is that albert-base-v1 ranks the relevant passage at the top place for 73.8% of total 164 questions and further fine-tuned version of this model with non-relevant class downsizing ranks the relevant passage at the top place for 75% questions outperforming any other models tried in this thesis. Although, albert-base-v1 is just behind the pre-trained model with a slight difference in the third decimal point in Mean-NDCG and MAP scores, with a smart selection of further fine-tuning dataset, it can be the best performing model when it comes to re-rank the relevant passage at the top place. Since these kind of re-ranking systems are mostly used for their performance at the top ranking, we can say that we achieve the best performing re-ranking approach with the smallest sized transformer model that we have tried.

In Table 6.5, a detailed analysis of the re-ranking performance is presented. This table can be interpreted as follows: the vertical numbers from 1 to 5 are the ranks of the relevant passage from the retrieval phase. The horizontal numbers from 1 to 5 denote the ranks of the relevant passage after the re-ranking phase using the model named further-fine-tuning-modified. Every number number in the table represents the transition in the ranking from the retrieval phase to re-ranking phase. For instance, there are 78 questions for which the correct passage was ranked in the first place after the retrieval phase stayed in the first place after the re-ranking phase. Similarly,

there are 21 questions for which the re-ranking algorithm moved the correct passages from the second place to the first place and there are 15 questions for which the correct passages were in the first place after the retrieval phase but these correct passages were moved to the second place after the re-ranking step. From this table we can conclude that although the re-ranking step may deteriorate rankings for a limited number of questions, it improves rankings for most of the questions. The first diagonal of this table represents the number of questions for which the ranking did not change after re-ranking phase. Numbers below the first diagonal of this table denote the improvements in the ranking results. Similarly, numbers above the first diagonal represent the deterioration in the ranking performance. It can be noted that the numbers below the first diagonal are usually bigger than the numbers above this diagonal. This is a natural result stemming from the improvements during the re-ranking phase utilizing the transformer models.

Table 6.4. Re-ranking Results catch@n Evaluations.

	Model	@1	@2	@3	@4	@5
1	**TF-IDF**	0.585	0.805	0.909	0.970	1.0
2	**ms-marco-MiniLM-L-12-v2**	0.720	0.927	0.970	**1.0**	1.0
3	**bert-base-multilingual-cased**	0.701	0.890	0.963	0.994	1.0
4	**bert-base-multilingual-uncased**	0.683	0.945	0.976	**1.0**	1.0
5	**distilbert-base-uncased**	0.671	0.884	0.957	0.994	1.0
6	**distilroberta-base**	0.671	0.896	0.951	0.982	1.0
7	**dstlbert-base-uncased (4 ep.)**	0.591	0.811	0.921	0.976	1.0
8	**distilroberta-base (4 ep.)**	0.634	0.866	0.957	0.994	1.0
9	**electra-small-discriminator**	0.695	0.902	0.963	0.988	1.0
10	**albert-base-v1**	0.738	0.872	0.957	0.994	1.0
11	**albert-base-v2**	0.707	0.909	0.976	0.994	1.0
12	**further-fine-tuning**	0.713	0.939	0.976	0.988	1.0
13	**further-fine-tuning-modified**	**0.75**	**0.951**	**0.994**	**1.0**	1.0

Table 6.5. Retrieval and Re-Ranking performance of further-fine-tuning-modified model.

Ret./Re-rank.	**1**	**2**	**3**	**4**	**5**
1	78	15	3	0	0
2	21	11	3	1	0
3	12	4	1	0	0
4	9	1	0	0	0
5	3	2	0	0	0

We can see a sorted comparative results for the models that have been tried in Figures 6.1 and 6.2. These figures summarizes the results explained in this section in terms of metrics mean-NDCG an MAP. In Figure 6.3, the catch@n counts (not ratios) can be investigated on each level including from catch@1 to catch@5. In Figure 6.4, a sorted comparison of model sizes can be seen.

6.3. Question Answering Results

As explained in Section 4.3, we have also re-purposed our best performing passage re-ranker model as a Question Answering module. In this third part of the Results Chapter, we give the evaluation of the performance for this approach. The evaluation for such a Question Answering pipeline is very similar to that of Machine Translation or Text Summarization We have reference answers for each question and hypothesis answer obtained by our model as described in Section 4.3.

There are three evaluations for the performance of this Question Answering pipeline: SQuAD evaluation script's F1 score, Rouge, and BLEU.

SQuAD evaluation script's F1 scores are presented in Table 6.6. Rouge evaluations are presented in Table 6.7. And similarly BLEU evaluations are presented in Table 6.8.

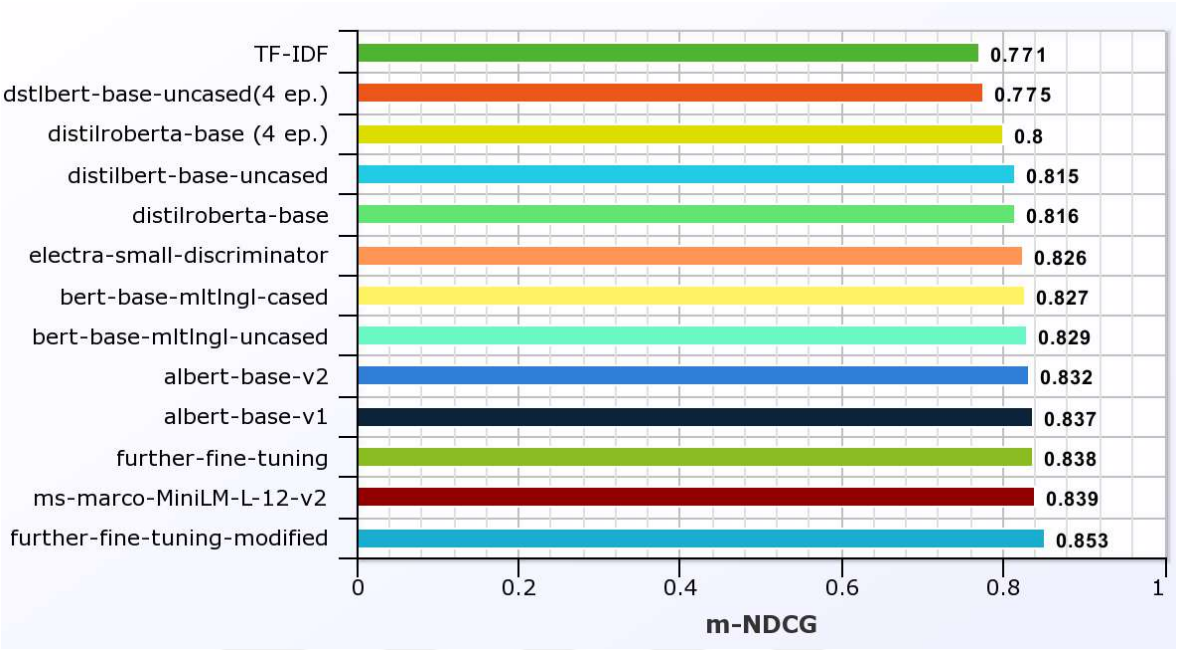

Figure 6.1. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, m-NDCG results.

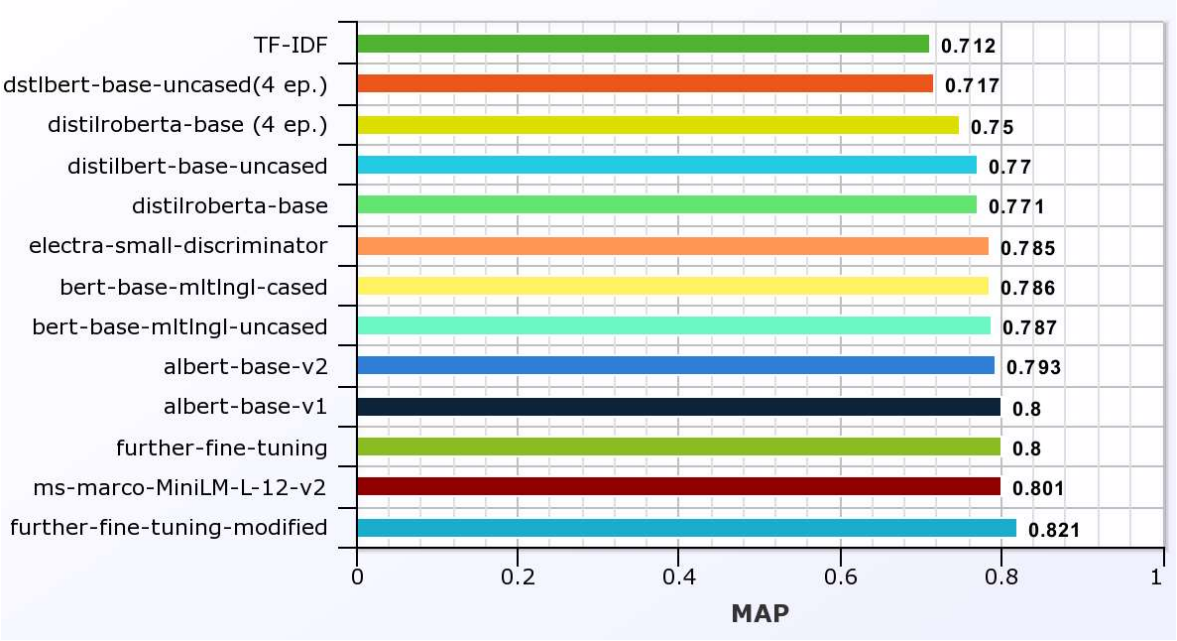

Figure 6.2. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, MAP results.

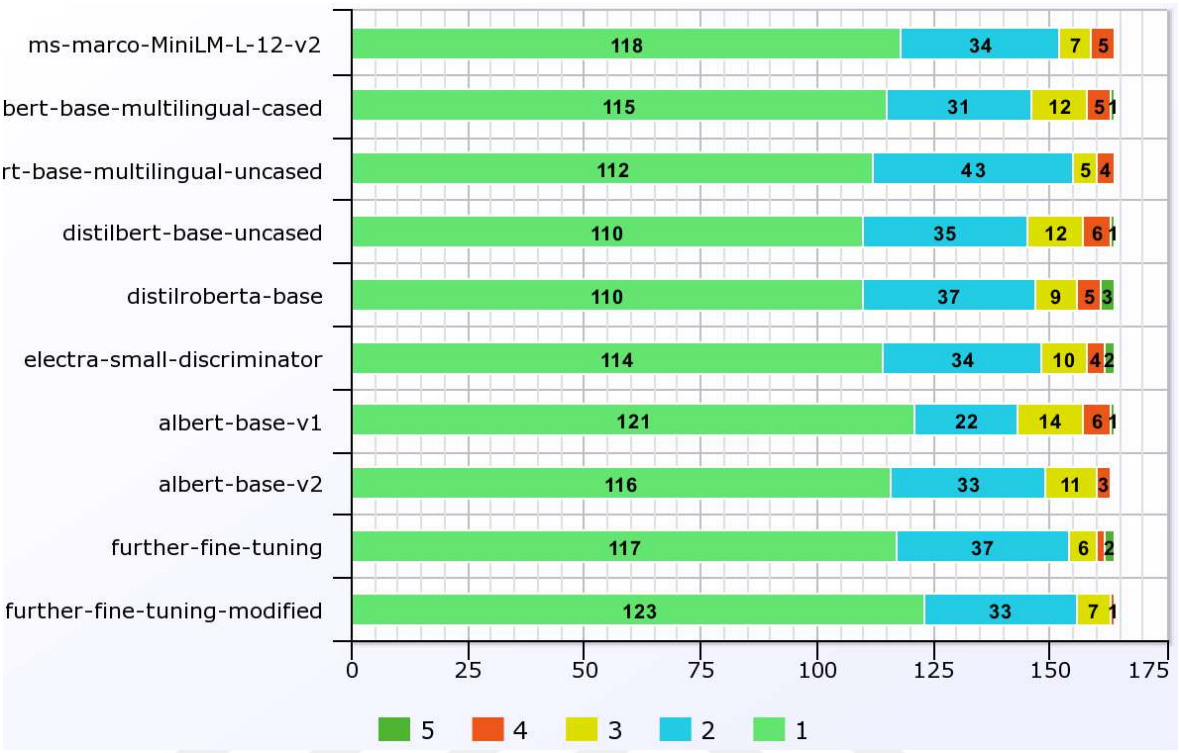

Figure 6.3. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, catch@n results.

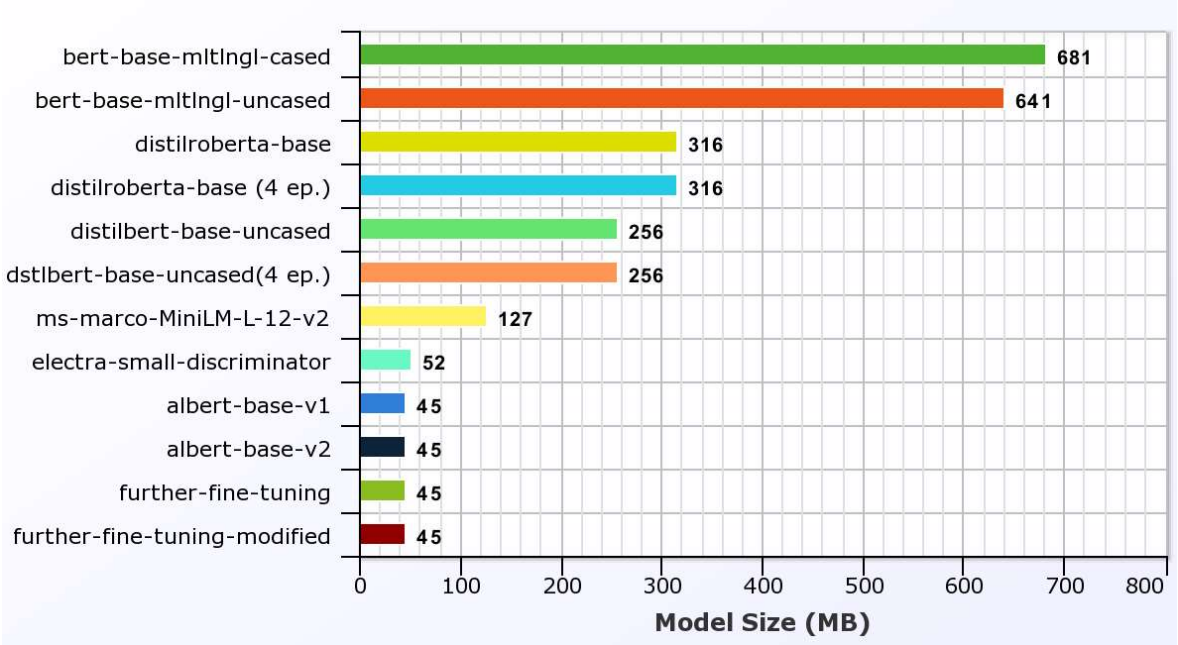

Figure 6.4. Our Re-ranking Implementation on Lecture Passage Re-ranking Dataset, Model Sizes.

We evaluated two models from our passage re-ranking trainings which are albert-base-v1 and further-fine-tuning-modified using SQuAD offical evaluation script as can be seen in Table 6.6. Since albert-base-v1 model performed better for this metric, we also evaluated it with Rouge and BLEU metrics.

Table 6.6. Question Answering Task SQuAD F1 Evaluation.

Model	**SQuAD F1**
albert-base-v1	49.96
further-fine-tuning-modified	41.27

In Table 6.6, the evaluations for the QA task that is described in Section 4.3 is presented in SQuAD official evaluation script's F1 metric. We have tried two models: albert-base-v1 and further-fine-tuning-modified. Since albert-base-v1 model performed better in this metric, we continued evaluations on this model with other metrics including Rouge and BLEU.

In Table 6.7, Rouge-n measures the number of matching n-grams between hypotheses and references. Rouge-l corresponds to Longest Common Subsequence (LCS) between hypotheses and references. The computation method for precision, recall, and F1 is given in the following formulas:

$$\text{precision} = \frac{\text{number of n-grams found in hypotheses and reference}}{\text{number of n-grams found in hypotheses}},$$

$$\text{recall} = \frac{\text{number of n-grams found in hypotheses and reference}}{\text{number of n-grams found in references}},$$

$$\text{F1} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}.$$

Table 6.7. Question Answering Task Rouge Evaluation.

Metric	Precision	Recall	F1
Rouge-1	0.733	0.592	0.564
Rouge-2	0.625	0.514	0.458
Rouge-l	0.705	0.565	0.540

In Table 6.7, the evaluations for the QA task is presented in Rouge metric. We see a performance of 0.564 F1 on Rouge-1 level, 0.458 on Rouge-2 level, and 0.540 on Rouge-l level. In their paper, Arısoy and Ünlü [5] mention that they observe a word level F1 score of 0.481 with MatchLSTM [49] and 0.467 with MReader [50] and outperform these baselines with 0.689 on the same Lecture QA Dataset. Here in our work, we achieve an F1 score of 0.4996 already outperforming the two baselines in the work of Arısoy and Ünlü [5]. They achieve an F1 score of 0.689 with BERT based methods; however, considering that the focus of our research has not been Question Answering and we don't make any specific modifications to our re-ranker system while re-purposing it towards the QA task, utilizing re-rankers in QA seems to be fertile research area.

In addition to Rouge evaluations of the model; in Table 6.8, BLEU evaluations for the same model is presented.

Table 6.8. Question Answering Task BLEU Evaluation.

BLEU score	**43.73**
Brevity Penalty	1.000
Hypothesis Length (characters)	4107
Reference Length (characters)	3893
Hyp/Ref Ratio	1.055

In Table 6.8, we also present BLEU scores for our model on the same task. BLEU is an algorithm to evaluate the quality of machine-translated text. Here, quality refers to the correspondence between an algorithms output and human translations.

Two examples from Lecture QA Dataset including passages, questions, answers, and predicted answers by albert-base-v1 model are shared in Tables 6.9 and 6.10.

Table 6.9. Sample 1 from Lecture QA Dataset (eval) [5].

Paragraph: signals are defined as functions of one or more independent variables that typically carry some type of information this figure shows a one dimensional signal here x axis is the time and y axis is the value of the function f of t for instance the y axis can represent the temperature or speed
Question: what is the definition of signals
True Answer: signals are defined as functions of one or more independent variables that typically carry some type of information **Predicted Answer:** signals are defined as functions of one or more independent variables

Table 6.10. Sample 2 from Lecture QA Dataset (eval) [5].

Paragraph: a discrete time complex exponential signal is defined as given here where capital c and alpha are in general complex numbers alternatively it is expressed as given below
Question: how can we define a discrete time complex exponential signal mathematically
True Answer: a discrete time complex exponential signal is defined as given here where capital c and alpha are in general complex numbers **Predicted Answer:** a discrete time complex exponential signal is defined as given here where capital c and alpha are in general complex numbers alternatively it is expressed

For Rouge metrics Rouge package and for BLEU evaluations sacrebleu package in Python programming language has been used. However; the important point is that independent of the implementation to evaluate scores, even with such a primitive application of re-ranking approach to the QA task, intriguing numeric evaluations have been achieved.

7. FUTURE WORK

The first improvement area for this work seems to be expanding transformer model pool that is evaluated. We have actually implemented the experiments for these new models including BERT-Base, RoBERTa-Base, xlnet-base-cased [51], xlm-roberta-base [52], Microsoft's deberta-base [53], roberta-large, bert-large-uncased, bert-large-cased, xlm-roberta-large, bert-multilingual-passage-reranking-msmarco and many others. However; because of the lack of enough computational resources, we couldn't try all these models.

Secondly, fine-tunings of the pre-trained Huggingface models have been done such that inputting query and passage in both query-passage and passage-query orders. Instead of this method, only query-passage order can be fed to the transformer networks and the size of the training dataset can be doubled by this way. Since the Passage Re-ranking task that is reseached in this work is actually an asymmetric semantic search, this new approach should improve the results.

The retrieval step can also be further improved. Since the TF-IDF structure retrieves the relevant document in top-5 ranking for 165 of all 173 questions, we did not see any urgency to improve the performance of the retrieval step. But, if the same approaches in this work are to be applied to larger collections of passages, documents, answers; it can be beneficial to improve the performance of the retrieval step. For instance, BM25 approach can be implemented towards this goal.

Another point that is open to improvement is that the domain knowledge of Lecture QA Dataset can be integrated to pre-training or fine-tuning processes of models. In Table 7.1, an example from Lecture QA Dataset is shown. As can be observed from this example, there are words and acronyms such as Laplace Transforms, Region of Convergence (ROC) in the passages and questions. If this domain knowledge can be trained to the transformer models, the model performance would improve further.

Table 7.1. A sample from Lecture QA Dataset (eval) including domain knowledge [5].

Paragraph: now lets look at the constraints on the region of convergence note that these constraints allow us to specify or to reconstruct the roc from knowledge of only the algebraic laplace transform expression and certain characteristics of the time domain function the first property states that the roc of x of s contains strips parallel to the j omega axis in the s plane this property can be easily verified by using the fact that the roc only depends on the real part of s the second property states that for rational laplace transforms the roc does not contain any pole note that x of s is infinite at pole locations therefore the laplace transform integral does not converge at a pole
Question 1: can the roc contain any poles **Answer 1:** for rational laplace transforms the roc does not contain any pole
Question 2: does the laplace transform converge at a pole **Answer 2:** the laplace transform integral does not converge at a pole

Lastly, the same approach can be applied to languages other than English with an easy switch to multilingual models existing in the Huggingface repository.

8. CONCLUSION

In this work, a number of experiments have been conducted for the tasks of Passage Retrieval, Passage Re-ranking, and Question Answering. In the first phase which is baseline experiments, effects of several hyperparameters such as query size, passage size, the size of the pre-trained BERT model, and batch size are investigated. After this step, an extensive study of transformer model size with respect to hidden size and number of layers has been done. It has been observed that clipping query and passages can have a negative impact on the model performance. Larger BERT models performed better in the baseline implementation. Batch size didn't have an effect on model performance between 32 and 4; but, when it is selected as 2, this change decreased the performance. When we have a test dataset leak into training data for passage re-ranking systems, this can create a positive bias in the performance of BERT models, even though this leak happened during pre-training phase indirectly. Also, it has been observed that for BERT models officially shared by Google through their Github repository, the effect of reducing hidden size has a bigger negative effect that reducing the number of layers of the model.

In the Passage Re-ranking experiments, we have achieved the same and even better performances with relatively smaller transformers compared to larger and slower models. We obtained best results with ALBERT's Base v1 model on our passage re-ranking task when we integrate a domain dataset in the fine-tuning process. This model outperformed eight other larger transformer networks.

We also re-purposed our best performing passage re-ranking transformer for a Question Answering task. Even without any study towards the QA task, we achieved comparable results that are open and curious to more research in the field.

REFERENCES

1. "SBERT, Training Overview", `https://www.sbert.net/docs/training/overview.html`, accessed in January 2022.

2. Reimers, N. and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, Association for Computational Linguistics, Hong Kong, China, Nov. 2019.

3. "SBERT, Cross-Encoders", `https://www.sbert.net/examples/applications/cross-encoder/README.html`, accessed in January 2022.

4. Nguyen, T., M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder and L. Deng, "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset", *Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*, Vol. 1773 of *CEUR Workshop Proceedings*, Barcelona, Spain, Dec. 2016.

5. Arısoy, E. and M. Ünlü, "Uncertainty-Aware Representations for Spoken Question Answering", *2021 IEEE Spoken Language Technology Workshop, SLT 2021*, Institute of Electrical and Electronics Engineers Inc., Shenzhen, China, Jan. 2021.

6. Mathur, V. and A. Singh, "The Rapidly Changing Landscape of Conversational Agents", *arXiv e-prints*, pp. arXiv–1803, 2018.

7. Hussain, S., O. Ameri Sianaki and N. Ababneh, "A Survey on Conversational Agents/Chatbots Classification and Design Techniques", L. Barolli, M. Takizawa, F. Xhafa and T. Enokido (Editors), *Web, Artificial Intelligence and Network Ap-*

plications, pp. 946–956, Springer International Publishing, Cham, Mar. 2019.

8. Robertson, S. and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond", *Foundations and Trends in Information Retrieval*, Vol. 3, No. 4, pp. 333–389, Jan. 2009.

9. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems", `https://www.tensorflow.org/`, accessed in January 2022.

10. "Huggingface", `https://huggingface.co/`, accessed in January 2022.

11. Guo, J., Y. Fan, Q. Ai and W. B. Croft, "A Deep Relevance Matching Model for Ad-Hoc Retrieval", *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, p. 55–64, Association for Computing Machinery, Indianapolis, Indiana, USA, Oct. 2016.

12. Xiong, C., Z. Dai, J. Callan, Z. Liu and R. Power, "End-to-End Neural Ad-Hoc Ranking with Kernel Pooling", *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, p. 55–64, Association for Computing Machinery, Tokyo, Japan, Aug. 2017.

13. Hui, K., A. Yates, K. Berberich and G. de Melo, "Co-PACRR: A Context-Aware Neural IR Model for Ad-Hoc Retrieval", *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, p. 279–287, Association for Computing Machinery, Marina Del Rey, CA, USA, Feb. 2018.

14. Mitra, B. and N. Craswell, "Duet at TREC 2019 Deep Learning Track", E. M. Voorhees and A. Ellis (Editors), *Proceedings of the Twenty-Eighth Text Retrieval Conference, TREC 2019*, Vol. 1250 of *NIST Special Publication*, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, Nov. 2019.

15. Dietz, L., M. Verma, F. Radlinski and N. Craswell, "TREC Complex Answer Retrieval Overview", E. M. Voorhees and A. Ellis (Editors), *Proceedings of The*

Twenty-Sixth Text Retrieval Conference, TREC 2017, Vol. 500-324 of *NIST Special Publication*, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, Nov. 2017.

16. Nogueira, R. and K. Cho, "Passage Re-ranking with BERT", *arXiv preprint arXiv:1901.04085*, 2019.

17. Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations", *International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020.

18. Clark, K., M.-T. Luong, Q. V. Le and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators", *International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020.

19. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *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)*, pp. 4171–4186, Association for Computational Linguistics, Minneapolis, Minnesota, USA, Jun. 2019.

20. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, "Attention is All You Need", *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6000–6010, Curran Associates Inc., Long Beach, California, USA, 2017.

21. Sanh, V., L. Debut, J. Chaumond and T. Wolf, "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter", *The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing, NeurIPS 2019*, Vancouver, Canada, Dec. 2019.

22. Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "Roberta: A Robustly Optimized BERT Pretraining Approach", *arXiv preprint arXiv:1907.11692*, 2019.

23. "Sentence Transformers: Multilingual Sentence, Paragraph, and Image Embeddings using BERT & Co.", `https://github.com/UKPLab/sentence-transformers`, accessed in January 2022.

24. May, C., A. Wang, S. Bordia, S. R. Bowman and R. Rudinger, "On Measuring Social Biases in Sentence Encoders", *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)*, pp. 622–628, Association for Computational Linguistics, Minneapolis, Minnesota, USA, June 2019.

25. Zhang, T., V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT", *International Conference on Learning Representations*, Addis Ababa, Ethiopia, Apr. 2020.

26. Qiao, Y., C. Xiong, Z. Liu and Z. Liu, "Understanding the Behaviors of BERT in Ranking", *arXiv preprint arXiv:1904.07531*, 2019.

27. Pennington, J., R. Socher and C. Manning, "GloVe: Global Vectors for Word Representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Association for Computational Linguistics, Doha, Qatar, Oct. 2014.

28. Conneau, A., D. Kiela, H. Schwenk, L. Barrault and A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data", *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Association for Computational Linguistics, Copenhagen, Denmark, Sep. 2017.

29. Cer, D., Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. St. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, B. Strope and R. Kurzweil, "Universal Sentence Encoder for English", *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 169–174, Association for Computational Linguistics, Brussels, Belgium, Nov. 2018.

30. Conneau, A. and D. Kiela, "SentEval: An Evaluation Toolkit for Universal Sentence Representations", *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, European Language Resources Association (ELRA), Miyazaki, Japan, May 2018.

31. Misra, A., B. Ecker and M. Walker, "Measuring the Similarity of Sentential Arguments in Dialogue", *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pp. 276–287, Association for Computational Linguistics, Los Angeles, USA, Sep. 2016.

32. Ein Dor, L., Y. Mass, A. Halfon, E. Venezian, I. Shnayderman, R. Aharonov and N. Slonim, "Learning Thematic Similarity Metric from Article Sections Using Triplet Networks", *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 49–54, Association for Computational Linguistics, Melbourne, Australia, Jul. 2018.

33. "MS MARCO, Microsoft Machine Reading Comprehension Dataset", `https://microsoft.github.io/msmarco/`, accessed in January 2022.

34. "TREC CAR; Text Retrieval Conference, Complex Answer Retrieval", `http://trec-car.cs.unh.edu/`, accessed in January 2022.

35. Dietz, L. and J. Foley, "TREC CAR Y3: Complex Answer Retrieval Overview", *In Proceedings of Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, USA, Nov. 2019.

36. Ünlü, M., E. Arisoy and M. Saraçlar, "Question Answering for Spoken Lecture Processing", *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, pp. 7365–7369, Brighton, UK, May 2019.

37. Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Association for Computational Linguistics, Austin, Texas, USA, Nov. 2016.

38. "dl4marco-bert, Passage Re-ranking with BERT", `https://github.com/nyu-dl/dl4marco-bert`, accessed in January 2022.

39. MacAvaney, S., A. Yates and K. Hui, "Contextualized PACRR for Complex Answer Retrieval", E. M. Voorhees and A. Ellis (Editors), *Proceedings of The Twenty-Sixth Text Retrieval Conference, TREC 2017*, Vol. 500-324 of *NIST Special Publication*, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA, Nov. 2017.

40. "google-research/bert", `https://github.com/google-research/bert`, accessed in January 2022.

41. "SBERT, SentenceTransformers Documentation", `https://www.sbert.net/index.html`, accessed in January 2022.

42. "PyTorch, An Open Source Machine Learning Framework That Accelerates the Path from Research Prototyping to Production Deployment", `https://pytorch.org/`, accessed in January 2022.

43. Goodfellow, I., Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

44. Schroff, F., D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering", *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, IEEE Computer Society, Los Alamitos,

CA, USA, Jun. 2015.

45. Bowman, S. R., G. Angeli, C. Potts and C. D. Manning, "A large annotated corpus for learning natural language inference", *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Association for Computational Linguistics, Lisbon, Portugal, Sep. 2015.

46. Williams, A., N. Nangia and S. Bowman, "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference", *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, Association for Computational Linguistics, New Orleans, Louisiana, USA, Jun. 2018.

47. "SQuAD, Stanford QUestion Answering Dataset", `https://rajpurkar.github.io/SQuAD-explorer/`, accessed in January 2022.

48. "Google Research, Colaboratory", `https://colab.research.google.com/`, accessed in January 2022.

49. Wang, S. and J. Jiang, "Machine Comprehension Using Match-LSTM and Answer Pointer", *International Conference on Learning Representations*, Toulon, France, 2017.

50. Hu, M., Y. Peng, Z. Huang, X. Qiu, F. Wei and M. Zhou, "Reinforced Mnemonic Reader for Machine Reading Comprehension", *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, p. 4099–4106, AAAI Press, Stockholm, Sweden, Jul. 2018.

51. Yang, Z., Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding", H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox and R. Garnett (Editors), *In Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS*

2019), Vol. 32, Curran Associates, Inc., Vancouver, Canada, Dec. 2019.

52. Conneau, A., K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale", *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Association for Computational Linguistics, Seattle, WA, USA, Jul. 2020.

53. He, P., X. Liu, J. Gao and W. Chen, "DeBERTa: Decoding-enhanced BERT with Disentangled Attention", *International Conference on Learning Representations*, Vienna, Austria, May 2021.

APPENDIX A: PERMISSIONS ABOUT FIGURE USAGE

In this thesis, "Figure 4.4. Bi-Encoder architecture with cosine similarity objective function" and "Figure 4.5. Bi-Encoder architecture with softmax classifier objective function" are re-drawn versions of "Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure)." and "Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function." from the paper "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks" by Nils Reimers and Iryna Gurevych, published in Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3982–3992, Hong Kong, China, November 3–7, 2019 [2]. This paper was published in EMNLP 2019 and it is accessible through ACL Anthology (https://aclanthology.org/D19-1410.pdf). The contents in ACL Anthology is open under Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

In addition to this, "Figure 4.3. Basic Training Architecture" and "Figure 4.6. Cross-Encoder Architecture" are re-drawn versions of figures from the website of the same publication (https://www.sbert.net/docs/training/overview.html and https://www.sbert.net/examples/applications/cross-encoder/README.html, respectively).

Plus, the authors of the same publication has been reached and their permission have been attained as can be seen in Figure A.1.