university of groningen

faculty of science and engineering

Parameter efficient transfer learning and fine-tuning of BERT-based transformers for Arabic QA

Nadir Al Hamzi

university of groningen

faculty of science and engineering

University of Groningen

Parameter Efficient Transfer Learning and Fine-tuning of BERT-based transformers for Arabic QA

Master's Thesis

To fulfill the requirements for the degree of
Master of Science in Artificial Intelligence
at University of Groningen under the supervision of
Dr. S.H. Mohades Kasaei (Artificial Intelligence, University of Groningen)
and
Dr. J.K. Spenader (Artificial Intelligence, University of Groningen)

Nadir Al Hamzi (s4180216)

March 7, 2022

Contents

Page

List of Figures **5**

List of Tables **6**

Acknowledgements **7**

1 Introduction **9**
1.1 Research Questions . . . 10
1.2 Thesis outline . . . 10

2 Background Literature **12**
2.1 Neural Networks . . . 12
2.1.1 Multi Layer Perceptrons . . . 12
2.1.2 Activation Functions . . . 13
2.1.3 Rectified Linear Unit . . . 13
2.1.4 Sigmoid . . . 14
2.1.5 Softmax . . . 14
2.1.6 Optimization . . . 15
2.2 Question Answering . . . 15
2.3 The Arabic language and challenges . . . 16
2.3.1 Arabic content in the internet . . . 16
2.3.2 Arabic orthography . . . 17
2.3.3 Arabic Morphology . . . 17
2.3.4 Challenges . . . 18
2.3.5 Studies in Arabic IR . . . 19
2.4 Passage Re-Ranking . . . 19
2.4.1 Classical IR . . . 20
2.4.2 Vector Space models . . . 20
2.4.3 Probabilistic retrieval models . . . 21
2.4.4 Research in PR . . . 21
2.5 Language Models . . . 22
2.5.1 BERT . . . 22
2.5.2 Pretraining of BERT . . . 22
2.5.3 AraBERT . . . 23
2.5.4 MARBERT and ARBERT . . . 23
2.5.5 Transformers rankings alternatives . . . 24
2.5.6 BERT in Arabic IR . . . 25
2.6 Transfer Learning . . . 25
2.6.1 Feature Based Transfer Learning . . . 25
2.6.2 Cross Lingual Transfer Learning . . . 26
2.6.3 QA Cross-Lingual Transfer Learning . . . 26
2.6.4 Multi-Task Transfer Learning . . . 27
2.6.5 Research in MTL . . . 27
2.6.6 Adapters . . . 28

2.6.7 AdapterHub 29

3 Methods **31**
3.1 Datasets 31
3.1.1 MS MARCO Dataset 31
3.1.2 Stanford Question Answering Dataset (SQuAD) 32
3.1.3 ARCD Dataset 32
3.1.4 TyDi QA Dataset 32
3.2 Tasks 32
3.2.1 BERT for Passage Ranking 32
3.2.2 BERT for MRC 33
3.3 Baselines 34
3.4 Preprocessing 34

4 Experimental Setup and Results **36**
4.1 Tools and Technologies 36
4.2 Proposed Models 36
4.3 Experimental Configurations and Hyperparameter Optimization 36
4.4 Evaluation Metrics 37
4.4.1 Mean Reciprocal Rank 38
4.4.2 Mean Average Precision 38
4.4.3 F1 score 39
4.4.4 Exact Match 39
4.5 Results 40
4.5.1 Experiment 1: Fine-Tuning on Passage Re-Rank Task 40
4.5.2 Experiment 2: Adapter-Tuning on Passage Re-Rank Task 40
4.5.3 Experiment 3: Fine-Tuning on Reading Comprehension Task 46
4.5.4 Experiment 4: Adapter-Tuning on Reading Comprehension Task 46

5 Conclusion **50**

Bibliography **52**

List of Figures

1 Proposed QA architecture with three modules: IR, re-ranker and reading comprehension. . . . 10

2 An example of a feedforward network with two inputs, two hidden unitts and one output unit. Source: [1] . . . 13

3 (a) Sigmoid function and (b) RelU and GELU functions. . . . 14

4 Most common languages used on the internet as of January 2020. Source:[2] . . . 17

5 Perfect Active, Passive Active, Perfect Passive and Imperfect Passive forms of the word

كتب. Source [3] . . . 18

6 BERT input for NSP task Source:Modified from original version in [4] . . . 22

7 BERT training architecture and fine-tuning. Source: [4] . . . 23

8 General architecture of ColBERT given a query q and a document d. Source: [5] . . . 25

9 Example architecture for multitask or transfer learning when the output variable y has the same semantics for all tasks while the input variable x has a different meaning. Source: [1] . . . 28

10 Dynamic customization possibilities where dashed lines in (a) show the current configuration options. These options include the placements of new weights (including down and up projections as well as new LayerNorms), residual connections, bottleneck sizes as well as activation functions.The resulting configurations for the architecture proposed by Pfeiffer et al.[6] and Houlsby et al. [7] are illustrated in (b) and (c) Source: [8] . . . 30

11 BERT encoder model for span-based question answering. Source: [9] . . . 34

12 results of fine-tuning mBERT and adapter tuning of mBERT evaluated on ARCD test dataset . . . 42

13 results of fine-tuning mBERT and adapter tuning of mBERT evaluated on TyDi QA test dataset . . . 43

14 results of fine-tuning ARBERT and adapter tuning of ARBERT evaluated on ARCD test dataset . . . 43

15 results of fine-tuning ARBERT and adapter tuning of ARBERT evaluated on TyDi QA test dataset . . . 43

16 results of fine-tuning MARBERT and adapter tuning of MARBERT evaluated on ARCD test dataset . . . 44

17 results of fine-tuning MARBERT and adapter tuning of MARBERT evaluated on TyDi QA test dataset . . . 44

18 results of fine-tuning AraBERT and adapter tuning of AraBERT evaluated on ARCD test dataset . . . 44

19 results of fine-tuning AraBERT and adapter tuning of AraBERT evaluated on TyDi QA test dataset . . . 45
20 results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta evaluated on ARCD test dataset . . . 45
21 results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta evaluated on TyDi QA test dataset . . . 45
22 results of fine-tuning mBERT and adapter tuning of mBERT (a) ARCD (b) Tydi QA 47
23 results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta (a) ARCD (b) Tydi QA . . . 48
24 results of fine-tuning ARBERT and adapter tuning of ARBERT (a) ARCD (b) Tydi QA 48
25 results of fine-tuning MARBERT and adapter tuning of MARBERT (a) ARCD (b) Tydi QA . . . 48
26 results of fine-tuning AraBERT and adapter tuning of AraBERT (a) ARCD (b) Tydi QA 49

List of Tables

1 Diacritic applied on the letter ف . . . 17
2 Sample records from Arabic MS MARCO dataset . . . 32
3 Question Answer Pairs from TyDi QA dataset . . . 35
4 Proposed models and associated number of languages and number of trainable parameters . . . 36
5 Hyperparameter Optimization used in fine-tuning the proposed models for passage re-ranking . . . 37
6 Hyperparameter Optimization used in fine-tuning the proposed models for MRC task 37
7 Hyperparameter Optimization used in adapter-tuning the proposed models for re-ranking task and machine reading comprehension . . . 38
8 Results on ARCD dataset, evaluated on the top 1000 retrieved passages. . . . 40
9 Results on TyDi QA dataset, evaluated on the top 100 retrieved passages. . . . 40
10 Adapter-tuning Results on ARCD dataset, evaluated on the top 1000 retrieved passages. 41
11 Adapter-tuning Results on TyDi QA dataset, evaluated on the top 100 retrieved passages. 41
12 Adapter-tuning impact on MRR@10, MRR@100, AVP@10 and AVP@100 compared to Fine-Tuning of the entire model on ARCD . . . 42
13 Adapter-tuning impact on MRR@10, MRR@100, AVP@10 and AVP@100 compared to Fine-Tuning of the entire model on TyDi QA . . . 42
14 Fine-Tuning results on ARCD dataset for the task of MRC. . . . 46
15 Fine-Tuning results on TyDi QA dataset for the task of MRC. . . . 46
16 Adapter-Tuning results on ARCD dataset for the task of MRC. . . . 47
17 Adapter-Tuning results on TyDi QA dataset for the task of MRC. . . . 47

Acknowledgments

First and foremost, I would like to thank my immediate supervisors, Dr. S.H. Mohades Kasaei and Dr. J.K. Spenader, to whom I could always turn for advice. Dr. S.H. Mohades Kasaei has supported me during my entire thesis and regularly provided valuable feedback. I would also like to show my deepest gratitude to Dr. J.K. Spenader, who guided and pointed me in the right direction for my thesis. I would also like to thank the High-Performance Computing team for their constant technical support.

Abstract

In today's digital era, Question Answering (QA) systems that automatically answer questions posed by humans in a natural language are gaining strong attention and becoming a model for the future of web search and conversational A.I. Unsupervised representation learning has significantly impacted natural language processing (NLP) from pre-trained embeddings to pre-trained contextual representations and transformer-based language models. Google's Bidirectional Encoder Representations from Transformers (BERT) models have shown superior results in QA tasks relying on large-scale, annotated datasets to achieve impressive performance. Although such datasets might exist for a few high-resource languages, many languages such as Arabic lack the labeled data needed to train deep neural networks from scratch. Progress on different aspects of Arabic Information Retrieval (IR) is severely lagging behind efforts in other languages. No recent study has examined the performance of different transformer-based models for Arabic-based QA systems. This study aims to establish different benchmarks for open-domain Arabic QA. It investigates a range of transfer learning strategies to address the challenges of training an IR-based QA system with three modules; retrieval, re-ranking, and machine reading comprehension (MRC). ElasticSearch is employed to index the collections for fast retrieval configured with the BM25 baseline for the initial stage. We evaluate parameter efficient transfer learning using adapters and fine-tuning transfer learning for the passage re-ranking and reading comprehension tasks. Monolingual and cross-lingual models, including mBERT, AraBERT, ARBERT, MARBER, and XLM-Roberta, are examined under two settings: adapter tuning and fine-tuning. Arabic Reading Comprehension Dataset (ARCD) and the Tydi QA development dataset are used for evaluation. The results reveal that monolingual models, in general, are superior to cross-lingual models. ARBERT achieves the highest score in passage re-ranking tasks and can serve as a baseline for re-ranking, while AraBERT offers a better benchmark for MRC. Furthermore, adapter tuning substantially reduces the model's size and speeds up training procedures without significantly impacting performance. It proves to be a viable alternative to traditional fine-tuning techniques.

1 Introduction

MRC is a challenging NLP task that has gained significant interest from researchers in recent years. The concept of MRC comes from the human understanding of the text and is one of the key problems in Natural Language Understanding [10]. The main goal of an MRC system is to find a relatively short answer to a query in an unstructured text. It makes it possible for systems to search through enormous amounts of data and deliver immediate answers to human queries in real-time. A wide range of real-world applications benefits from such systems as search engines, dialog systems, and information retrieval (IR). The remarkable progress of MRC in recent years is mainly driven by the availability of large-scale datasets coupled with the emergence of large pre-trained language models such as Google's Bidirectional Encoder Representations from Transformers models [4].

Up until now, most advances in NLP and efforts in building large, annotated datasets have been focused on English. Other languages, such as Arabic, lack equivalent resources despite being one of the largest spoken languages in the world. It has received comparatively little attention in modern computational linguistics. Nevertheless, there has been considerable effort in recent years toward building high-quality datasets to leverage the training of language models [11]. Transfer learning from large language models has gained widespread attention from researchers, and we see the adaptation of google transformer models for many NLP/NLU tasks. Following the success of BERT on many NLP/NLU tasks on English datasets, we see several Arabic BERT-based models being published in the last three years. However, fine-tuning large language models consisting of millions or billions of parameters requires a considerable amount of training. As a result, progress towards more general and versatile NLP methods that can be applied to many tasks is hindered. A more efficient parameter transfer learning alternative is proposed in [12], which adds new task-specific parameters called adapters to the model. In this setting, the newly added parameters are trained during the fine-tuning phase while the original model parameters are fixed. With adapter modules, fine-tuning becomes very parameter efficient, and the same original model can be shared between all downstream tasks. Adapters have also proven to perform on par with fine-tuned models, especially in low-resource languages.

To the best of our knowledge, no recent comprehensive study evaluates the different state-of-the-art transformers-based models for passage re-ranking and reading comprehension for the Arabic language. Among the current work in Arabic IR is the work of Mozannar, Hajj, and Hajalin, in which a hierarchical tf-idf is used in the retrieval model, and a multi-lingual BERT-based model is fine-tuned for reading comprehension tasks [13]. As a baseline for the retrieval model, the study used Wikipedia API search and Google custom search API performing low compared to the proposed hierarchical tf-idf. Other studies such as in [14] evaluated different models on various NLP tasks, including reading comprehension but not in re-ranking tasks.

To further tackle the challenge of Arabic QA and to spur research in Arabic IR, we propose a comprehensive study that extends the previous work of [13] to include the latest pre-trained BERT models in both the retrieval stage and reading comprehension stage. A full QA pipeline is proposed as part of the study, including an IR module, a re-ranker module, and a reading comprehension module. ElasticSearch is employed to index the collections for fast retrieval configured with the BM25 baseline for the initial stage rather than the hierarchical tf-idf baseline used in [13]. The re-ranking module is responsible for further re-ranking the documents based on query relevancy. At the same time, the

reading comprehension module is responsible for extracting the answer span given a candidate passage and a query. We evaluate SOTA models including mBERT, AraBERT, ARBERT, MARBER and XLM-Roberta for the re-ranking and MRC modules under two settings: adapter tuning and fine-tuning. Parameter-efficient tuning is motivated by the fact that the parameters of the original network are frozen and may be shared by the two tasks. The goal is to achieve the same performance as fine-tuning but with fewer trainable parameters. The proposed architecture is best described in Figure 1. Similar to the work of [13] and [14], the ARCD test dataset is used for evaluation, and in addition, we include Arabic TyDi QA development set. To enhance the assessment of the passage re-ranking task, we also include the mean average precision and mean reciprocal rank metrics. For MRC, the models are evaluated using F1 metric and Exact Match as reported in the work of [14].

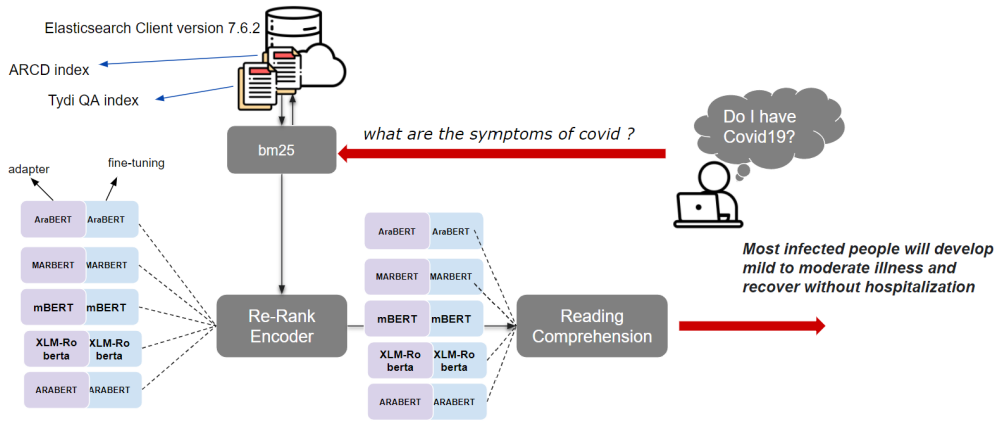


Figure 1: Proposed QA architecture with three modules: IR, re-ranker and reading comprehension.

1.1 Research Questions

To summarize, this thesis focuses on the following problems:

Q1. How does monolingual transfer learning perform in a re-ranking task in comparison with the baseline ranker BM25 and to cross-lingual transfer learning?

Q2. How effective is monolingual transfer learning in reading comprehension tasks for the Arabic language compared to cross-lingual transfer learning?

Q3. Can parameter efficient transfer learning match the performance of fine-tuning entire models for low resource languages such as Arabic in QA tasks?

Q4. How does parameter efficient transfer learning perform when applied on monolingual models compared to cross-lingual models ?

1.2 Thesis outline

The structure of the thesis is as follows: Chapter 2 provides the theoretical background relevant to the research, including neural networks, question answering, Arabic language, language models, and transfer learning. Chapter 3 describes the methodology used to carry out the experiments, including the datasets used for training and evaluation and a detailed description of each task and baselines.

Chapter 4 describes the setup and configurations of the different tools used in the methods. In Chapter 4, we also present the results of the experiments. Finally, in Chapter 5, we interpret the results in light of the research questions and provide the answers to all questions.

2 Background Literature

This chapter introduces the theoretical framework underlying the current research and covers topics that are relevant to the study. The first section explains topics in neural network which are the backbone of BERT transformers. The following section describes the history of Question Answering and its relevance to AI. Section 3 is dedicated to the Arabic language and provides an overview of the Arabic morphology and orthography and relevant research in Arabic IR. Section 4 discusses different IR systems and relevant research in passage re-ranking. Section 5 introduces the SOTA language models that are used in the research. The last section describes the different transfer learning techniques which includes feature based transfer learning, cross lingual transfer learning and parameter efficient transfer learning.

2.1 Neural Networks

Inspired by the mechanism of learning in biological organisms, artificial neural networks are among the most advanced machine learning techniques and are often referred to as neural networks. The cells in a human nervous system contains neurons, highly specialized nerve cells responsible for communicating information and are considered the building blocks of the nervous system. These nerver cells are interconnected via axon and dendrites, specialized structures designed to transmit and receive information. Communication between two neurons is achieved through a synapse a small gap between neurons. The strength of this connection often change in response to external stimuli. This change is how learning takes place in living organisms.

2.1.1 Multi Layer Perceptrons

Multi-Layer Perceptrons (MLP) are interconnected computational nodes typically aggregated into networks of layers composing many different functions. In MLP, which are also called deep feedforward networks or feedforward networks, information flows from the input layer across the middle layers to the output layer. Mathematically, A feedforward network maps an input x to class y where the goal is to approximate some mapping f^* such that

$$y = f(x;\theta).$$

The model is usually associated with a directed acyclic graph that defines how the network's underlying functions are composed [1]. Figure 2 shows an example of a simple feedforward network with two hidden units. The hidden vector h is computed by a function

$$h = f^{(1)}(x;W,c).$$

The hidden vector h is then passed to the output layer, simply a linear regression model given by

$$y = [f^{(2)}(h;w,b)$$

. The full network computation is presented by

$$f(x;W,c,w,b) = f^{(2)}(f^{(1)}(x)).$$

The function $f^{(2)}$ is called an activation function and is typically modeled as a non linear function.

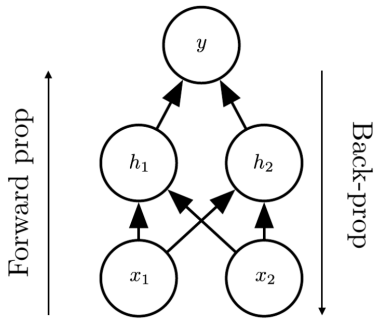


Figure 2: An example of a feedforward network with two inputs, two hidden unitts and one output unit. Source: [1]

2.1.2 Activation Functions

In neural networks, activation functions are generally non-linear functions that compute the hidden layer values. The design of activation functions is an active area of research, and common choices are differentiable and easy to optimize non-linear functions. Models with non-linear functions can better capture complex data patterns and differentiate between outcomes. When applied to linear inputs, non-linear activation functions yield non-linear transformation, increasing the model's capacity. Almost all neural networks are modeled with non-linear activation functions.

2.1.3 Rectified Linear Unit

Rectified Linear Unit (RELU) is a monotonic increasing function that ranges from 0 to infinity and is given by:

$$g(z) = max(0, z).$$

RELU is differentiable everywhere except at 0. It is easy to optimize and does not introduce second-order effects. The rectified linear unit behaves as a linear function except that a rectified linear unit outputs zero across half its domain making it's derivative large whenever the unit is active. It is presumably the most commonly used activation function in neural networks, especially in convolution neural networks and deep neural networks. There are many generalizations of RELU based on using a nonzero slope α_i when $x_i \leq 0$

$$h_i = g(z, \alpha)_i = max(0, z_i) + \alpha_i min(0, z_i)$$

1. Gaussian Error Linear Unit (GELU): A smooth approximation to the rectifier

2. Absolute value rectification: sets $\alpha_i = -1$ to obtain g(z) = $|z|$, it can seek features that are invariant under a polarity reversal.

3. A Leaky ReLU fixes α_i to a small value like 0.01

4. The parametric ReLU, or PReLU treats α_i as a learnable parameter.

2.1.4 Sigmoid

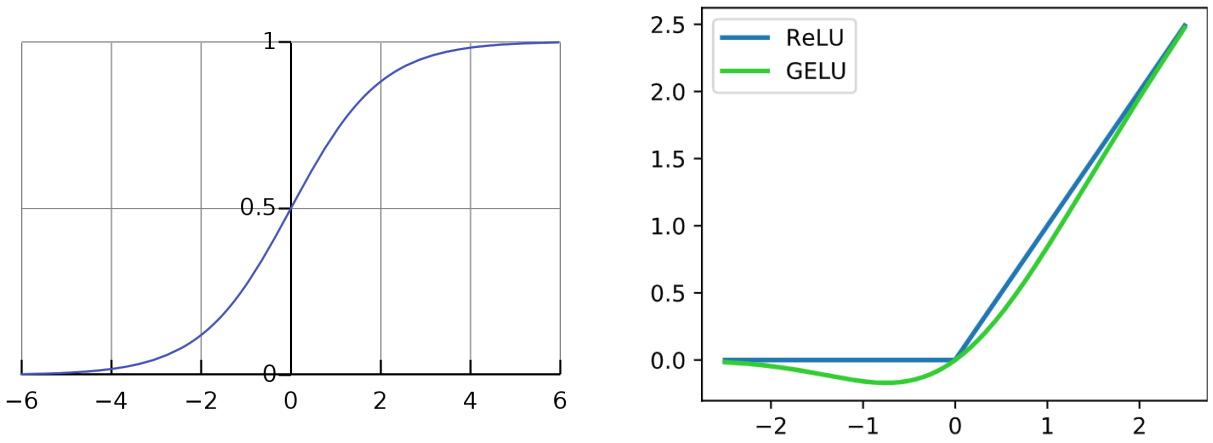
Sigmoid function also known as the logistic activation function is a monotonic function with S-shape curve with outputs ranging from 0 to 1 and given by:

$$\sigma(x) = \frac{1}{1+\exp^{-x}}$$

The sigmoid output is suitable for binary classification where only one unit is needed to predict the

$$P(y=1|x)$$

The sigmoid derivative is not monotonic, and the function can saturate with high input values. Figure 3 shows the outputs of sigmoid, RelU and GELU activation functions.
[15]



(a) Sigmoid function. Source: [15]

(b) Relu and Gelu functions. Source:[16]

Figure 3: (a) Sigmoid function and (b) RelU and GELU functions.

2.1.5 Softmax

We can use the softmax function to represent a probability distribution over a discrete variable with n classes. In the first step, a linear layer predicts the unnormalized log probabilities:

$$z = Wh + b$$

where z_i is $P^{'} = (y = i|x)$. To obtain the desired $y^{'}$, the softmax function exponentiates and normalizes z:

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j \exp(z_j)}$$

The softmax function is a generalization of the sigmoid function for multilabel classifications.

2.1.6 Optimization

Optimization plays an essential role in many aspects of deep learning algorithms. The most challenging optimization problem in deep learning is neural network training, where the objective is to reduce the error over the training set, such as:

$$J(\theta) = E_{(x,y)\ P^*_{data}} L(f(x;\theta), y)$$

where L is a loss function, p^*_{data} is the empirical distribution, $f(x;\theta)$ is the predicted output, and y is the truth output [1]. Although minimizing the true objective function by taking the expectation over the true data generating distribution is preferable is it's unlikely that the p_{data} is known. In machine learning, we convert the problem into an optimization problem by replacing the true distribution $p(x,y)$ with the empirical distribution p^*_{data} defined by the training set. The objective is to minimize the empirical risk given by:

$$E_{x,y\ P_{data}}[Lf(x;\theta), y)] = \frac{1}{m}\sum_{i=i}^{m} Lf(x^i;\theta), y^i) \tag{1}$$

[1]

2.2 Question Answering

Designing an intelligent QA system that surpasses a human's level in an open domain is one of the oldest goals of artificial intelligence and dates to the 1960s. The main goal of a QA system is to find an answer to a query from a large collection of data (structured or unstructured). Early QA systems focused on closed domains in which the system could answer only domain-specific questions. BASEBALL, one of the earliest and successful QA systems developed in the 1960s, answered questions formed in natural language about the U.S baseball league over a year [17]. For example, it can answer questions like:

- Which team lost on May 3
- Where was the match held?
- Who won the match?

LUNAR, another well-known QA system, covered questions about the geological analysis of rocks returned by the Apollo moon mission [18]. It was demonstrated in a science convention in the 1970s to answer different questions related to the analysis of rocks, such as:

- What is the specific activity of A126 in soil?
- What are the analyses of strontium in plagioclase.
- What are the plag analyses for breccias?

As it turns out, LUNAR could answer 90% of questions in its domain posed by people who had no training with it. Each system focused on answering questions within a narrow domain, and both performed well inside their defined scenario. Following that, other restricted-domain QA systems were developed, all of which had a knowledge base or database compiled by experts in their chosen

domains.

QA systems are classified into closed domain QA systems and open-domain systems. Open-domain systems aim to answer a question in the form of natural language based on large-scale unstructured documents such as Wikipedia. Over the past years, there has been a notable increase in research publications on Open-domain systems focusing on information retrieval and techniques that integrate with neural Machine Reading Comprehension (MRC). In literature, open-domain QA has been studied closely with research in Natural Language Processing (NLP), Information Retrieval (IR), and Information Extraction (IE). A core goal in artificial intelligence is to build systems that can sift through large unstructured and structured data and then answer complex questions about any subject. Querying information from a large corpus has become very important in today's digital age. There is a lot of textual data, FAQs, newspapers, articles, documentation, user cases, customer service requests, etc. Quite hard to keep in mind all that information. Who won the last football Euro Cup? What is Bitcoin? To find some information, we need to search a document and spend a couple of minutes reading it before finding the answer. It is estimated that about 2.3 trillion gigabytes of data are created each day, and the global amount of the data created, consumed or copied reached over 64 zettabytes in 2020 [19]. Most companies in the U.S. only have at least 100 terabytes of data stored.

Due to its practical application in search engines, intelligent support systems, conversational AI, and numerous other commercial applications, tech giants like Microsoft and Google invest heavily in language understanding research to provide the user with a better search experience. Recently a team of scientists at DAMO Academy, Alibaba's global research program, developed a QA system that outperformed human scores on Microsoft's Machine Reading Comprehension dataset, one of the most challenging reading comprehension tests in artificial intelligence [20]. It scored 0.54 in the MS Marco question-answering task, outperforming the human score of 0.539, a benchmark provided by Microsoft [20].

2.3 The Arabic language and challenges

Arabic is a language that was descended from Semitic languages, and it is also the language of the Quran, making it an important heritage language [21]. There are three forms of the Arabic language: Quranic or Classical Arabic, which is the language of Quran, Modern Standard Arabic (MSA), and Colloquial Arabic. The Quranic Arabic is mainly used in reciting the Quran or quoting older religious texts. In the 7th century, following the spread of Islam to different parts of the world, the Arabic language spread to the Balkans, Egypt, and North Africa by dominating these regions [22]. Today, Arabic is spoken by about 420 million people around the world. Many languages like Spanish, Turkish, Persian, Urdu, Swahili, and Hausa, mainly in terms of vocabulary, were influenced by Arabic. Other countries such as Eritrea, Chad, and Somalia also have Arabic as the country's official language [22]. Modern Standard Arabic (MSA) is considered the official language of the Arab world, including Northern Africa, the Arabian Peninsula, and Mesopotamia. Like classical Latin, MSA is used in literature, academia, print and mass media, law, and legislation, but it is generally not spoken as a first language.

2.3.1 Arabic content in the internet

As more language communities become connected to the internet, the internet has become a multilingual environment that expands in diversity. More than half of the homepages of most websites on

the World Wide Web are in English, with varying amounts of information included in other languages such as Arabic. Based on a study by W2Techs, Arabic is expected to constitute 1.2% of the top 10 million websites as of November 2021 [23], [24]. It is estimated that around 5.2% of all internet users are Arabic speakers, which ranks 4th after English, Chinese, and Spanish as shown in Figure 4 [2].

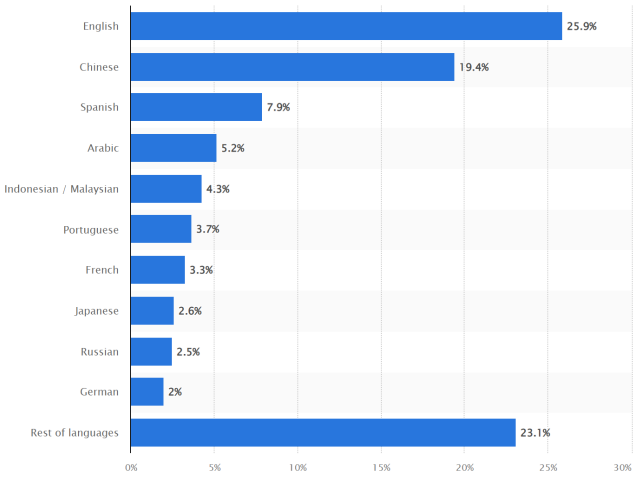


Figure 4: Most common languages used on the internet as of January 2020. Source:[2]

2.3.2 Arabic orthography

The Arabic alphabet consists of 28 letters in total and has only three vowels ا (Alef), و (waw) and ي (ya). The shape of a character can differ based on its location within a word. For example, the letter ج pronounced jeem is written as جـ if it is at the beginning of the word as in جميلة meaning beautiful. On the other hand, it is written as ج if it is placed at the end of the word as in مزاج which means mood. The Arabic orthography lacks short vowels letters but includes a set of marks, called diacritics, that are placed above or below the letters. The main purpose of diacritics is to provide a phonetic guide showing the correct pronunciation of the words [25]. Table 1 shows the four basic diacritics applied on the letter ف and the correct pronunciation of the letter with each diacritic.

Diacritic:	skon	fat-ha	dhama	kas-ra
Letter + Diacritic:	فْ	فَ	فُ	فِ
Pronunciation:	f	fa	fo	fe

Table 1: Diacritic applied on the letter ف

2.3.3 Arabic Morphology

In Arabic, nouns and verbs are derived using a strict morphophonemic rule, making Arabic a highly derived language. Verbs stem from the 3, 4 base words, while nouns, including adjectives and adverbs, stem from 3,4,

or 5 base letters. Derivation mainly relies on templates and patterns to identify the root of a verb or noun. It is also possible that these base letters appear with extra letters or be modified due to morphophonemic rules based on surrounding letters and vowels. The word write كتب in Arabic can have more than 20 forms based on the context as illustrated In Figure 5

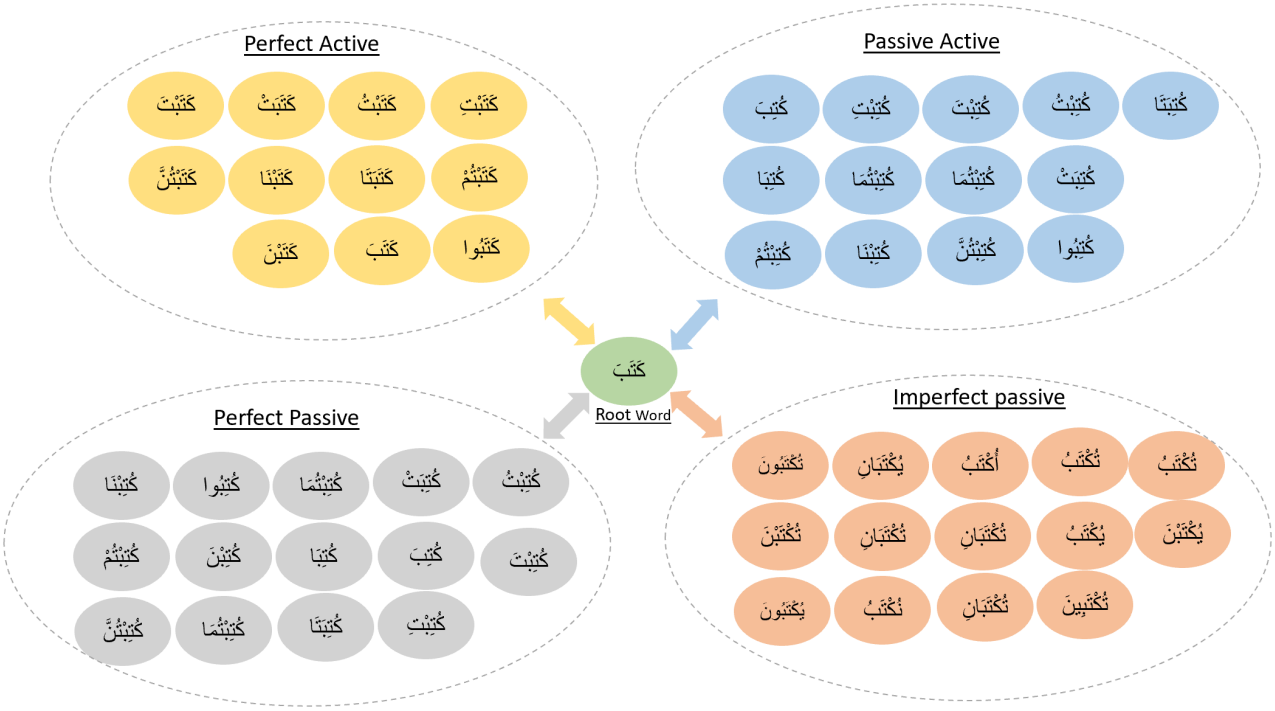


Figure 5: Perfect Active, Passive Active, Perfect Passive and Imperfect Passive forms of the word كتب. Source [3]

2.3.4 Challenges

The Arabic language is highly expressive, and the Arabic lexicon is rich in vocabulary and synonyms. Like English, each word can have a different meaning based on its context. Moreover, Arabic dialects are spoken in many regions without a standardized written form. It is also possible to find different dialects within the same country, but the most significant variations are between regional language groups. Dialectal words may have other morphological forms than MSA. The word عيش for example, means living in MSA but in Gulf Arabic, it means rice while in Egyptian Arabic عيش means bread. There are around 12 different dialects, some rarely used or extinct, such as Andalusi Arabic [26]. With the spread of online social interaction within the Arab world, dialects are surfacing the online world, especially in Social Media Forums, Discussions, and Message Boards. Arabic lexicon, dialect variations, and lack of uniformity in writing styles create additional challenges

for sentiment analysis, hate speech detection, and similar tasks. Applications such as text summarization and machine translation may also be hindered by the presence of multiple-word expressions in the Arabic language and lack of consistency in orthography, nonappearance of capital letters and inherent ambiguity in named entities [27].

2.3.5 Studies in Arabic IR

In the early 1990s, most studies on Arabic IR relied on classical approaches based on tests that contained a few hundred documents with Arabic encoded characters [27]. The early 2000's witnessed a growing interest in Arabic processing and retrieval driven mainly by the Text REtrieval Conference (TREC) [28]. TREC aims at supporting research within the information retrieval community by providing the infrastructure to evaluate text retrieval methodologies on a large scale. The conference is co-sponsored by U.S [29]. Department of Defense and the National Institute of Standards and Technology (NIST)[28]. The TREC workshop series aims to accomplish the following goals:

1. Advance research on information retrieval based on large test collections.

2. Establishing an open forum for the exchange of research ideas among industry, academia, and government;

3. Organization of Annual IR Research Workshops and Development of Standardized IR Evaluation Methods. To provide industry and academia with proper evaluation techniques and new systems more compatible with current evaluation techniques.

Several other IR evaluation campaigns were started following TREC, such as CLEF, INEX, NTCIR, and FIRE, to help improve IR systems for different languages and domains [30]. Due to the absence of IR evaluation campaigns specifically for Arabic, fewer shared tasks are available in campaigns that focus on Arabic. However, there were also tasks in these campaigns that investigated Arabic as a target language, such as the TREC CLIR in 2001/2002 and the CLEF INFILE in 2008/2009, which spurred research toward Arabic IR [31], [32].

2.4 Passage Re-Ranking

A typical MRC architecture consists of a retrieval stage, a re-rank stage, and a document reader stage. The IR task, often referred to as a search engine, is the process of finding relevant material from a large collection of documents (corpus) that satisfies a particular information need (query) that is unstructured in nature (usually text). As defined in this way, information retrieval used to be an activity that only a few people engaged in: reference librarians, paralegals, and similar professional searchers. Now the world has changed, and hundreds of millions of people engage in information retrieval every day from multiple devices, including smartphones. In general, there are three different types of IR (usually distinguished by scale)

1. Web Search: Billions of documents over millions of machines

2. Personal IR: Mac OS Spotlight / Windows Vista's Instant Search

3. Enterprise, Institutional, Domain-Specific Search: Corporation's internal documents, a database of patents, research articles, etc.

Other areas of IR include Clustering, in which documents are grouped based on content, and Classification, where records are classified based on their content. We assume a (potentially large) corpus D that users might wish to search as a standard formula

2.4.1 Classical IR

The Boolean Retrieval (BR) model is perhaps the simplest IR model in which a bag of words represents the documents, and queries are expressed in Boolean. The documents consist of terms that represent pieces of information in the document, which are linguistically preprocessed to produce normalized tokens representing the word's index. An essential element in the BR model is the term-document incidence matrix, a sparse matrix where the columns represent the documents and the rows represent the terms. The matrix $element_{i,j}$ is 1 if the term at row i appears in $document_j$. Computing these indexes is time-consuming and computations happen offline so that only the precompiled indexes are accessed during runtime. The queries are expressed using Boolean expressions in which search terms can be combined with AND, OR, and NOT operators as a filtering mechanism. A bitwise operation is then applied between the query and all document vectors to retrieve relevant results. Mathematically this can be described as:

Let

$$Tk = \{t_k1, t_k2, ..tk_n\}$$

be a set of normalized token. A document is then a subset of these tokens. Let

$$D = \{D_1, D_2, ..D_m\}$$

be set of all m documents. A query Q is a Boolean expression of the form

$$Q = (W_1 \vee W_2 \vee \cdots) \wedge \cdots \wedge (W_i \vee W_{i+1} \vee \cdots)$$

where W_i is true for D_j when t_i in D_j [33].

Despite its speed and simplicity, Boolean retrieval suffers from many drawbacks. The term-document matrix is very sparse and memory inefficient. Boolean retrieval also lacks the support for more complex queries and is incapable of similarity search.

2.4.2 Vector Space models

Traditional IR systems use a vector space model, in which queries are converted into vectors based on their unigram word counts. The documents are then ranked based on the cosine similarity measure with the query vector as in 2.

$$\cos\theta = \frac{\vec{q}.\vec{D}}{||\vec{q}|| \,.\, ||\vec{D}||} \tag{2}$$

There are two commonly used weighting schemes: tf-idf and a slightly more powerful variant named BM25 [34], [35]. Tf-idf is a measure of how important a word is to a document in a collection. It is governed by the term frequency tf and inverse document frequency idf. The term frequency tells us how frequent the word is within a document d and is given by

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

(3)

where $f_{t,d}$ is the number of times the term t appears in d [36]. On the other hand, the IDF term weights the terms based on how much information it adds to a document and can be obtained by taking the logarithmically scaled inverse of the total number of documents N over the number of documents that contain the term n_t given by:

$$idf(t,d) = -log\frac{n_t}{N}. \tag{4}$$

The log here is used to dampen the importance of a term so that the relevance of a term t does not increase proportionally with term (or document) frequency. Tf-idf is then the product of the above two terms leading to:

$$\text{tf-idf} = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} . - log \frac{n_t}{N} .$$

(5)

Terms that occur in a handful of records are useful for discriminating those documents from the rest of the collection; terms that occur across the entire collection aren't as helpful.

2.4.3 Probabilistic retrieval models

First developed in 1995, BM25 is considered one of the most successful models for text retrieval [35]. It is a probabilistic retrieval model based on the best match (BM) family that looks into the term frequency. BM25 is viewed as an enhancement to the traditional tf-idf weighing scheme. In contrast to tf-idf which penalizes frequent terms, BM25 takes term frequency and document length into account. It adds two additional terms, k a parameter that adjusts the balance between the term frequency and inverse document frequency, and b, which controls the importance of document length normalization. The BM25 score of a document d given a query q is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) . \frac{f(q_i,D).(k_1+1)}{f(q_i,D) + k_1.(1-b+b.\frac{|D|}{avgdl})} \tag{6}$$

where $f(q_i,D)$ is the frequency of q_i in document D, $\mid D \mid$ is the length of the document D , and avgdl is the average document length in the collection. [37].
One shortcoming of the model is the absence of instructions for setting these internal parameters, impacting the final scoring. However, it does leave an opportunity for optimization based on experimentation and human judgment. Nevertheless, empirical results suggests values such as

$$0.5 < b < 0.8$$

and

$$1.2 < k1 < 2$$

provide the best results. [38]. Over the past years, other variations of BM25 have been developed by adding new components like (k1 + 1) to the numerator of the function, changing specific parameters, or eliminating certain terms.

2.4.4 Research in PR

In the field of IR, previous models have shown great promise, and BM25, in particular, provides a solid framework for evaluating newly developed methods. More computationally intensive methods are usually used to rank the returned document following the IR stage. Intuitively, this stage directly impacts the performance of the reading comprehension module responsible for extracting the final answer. When searching in Bing or Google, we quickly skim the first page and select only a handful to read, but it is far more difficult for a machine. Machines, however, can use built-in heuristic rules or learn statistical models from large text corpora to provide better results. An excellent way to improve the ranking results is to eliminate candidate answers clearly out of the ordinary by reasoning about the input query. The authors in [39] looked at parts of the syntactic structures of the question and passages as a candidate-answer check. A question's type dictates the expected answer type. For example, "How many states in the U.S.A?" has an answer type number, whereas "Where is

Groningen University located?" has the answer type location. The re-ranker's task is to estimate how relevant a candidate passage $d_i$is to a query q. Only passages passing the candidate-answer check are passed to the reading comprehension stage. The model has achieved high-performance results on three public open-domain QA datasets: Quasar-T, SearchQA, and the open-domain version of TriviaQA.

However, neural approaches have been extensively researched in the field of NLP and IR, outperforming all previous methods. In the following sections, I introduce BERT transformers and state-of-the-art techniques in PR.

2.5 Language Models

2.5.1 BERT

BERT is a language model that belongs to the transformer-based family of models, first introduced in 2019 by the Google AI team [4]. The model was shown to achieve state-of-the-art accuracy on many NLP and NLU tasks, including the QA task on SQuAD v1.1 and v2.0. Transformer architectures rely on encoder-decoder networks that use self-attention and attention mechanisms on the encoder and decoder sides, respectively. BERT is essentially a stack of transformers that follows the architecture described in [40] and can behave as an encoder as well as a decoder. There are two variants of BERT models: BERT Base: with only 12 layers in the encoder stack, 12 attention heads, and 110 million parameters. BERT Large: with 24 layers in the encoder stack, 16 attention heads, and 340 million parameters [4].

2.5.2 Pretraining of BERT

BERT is trained using WordPiece tokenization, which further breaks the words down into sub tokens and expects the input sequence to be of specific format. Special tokens are added to the input [CLS] and [SEP] for the model to understand the input sequence better. [SEP] token helps the model recognize the end of one sentence and the start of another sentence in the same sequence. The [CLS] token, which stands for classification, contains the last hidden state of the BERT h([CLS]), which summarizes BERT understanding of the entire sequence and is used as input for classification tasks. In addition to word tokens, BERT uses positional embeddings and segment embeddings as shown in Figure 6. With BERT, the input is processed simultaneously, and in the absence of any recurrent states, the word order information is provided as positional encodings [4].

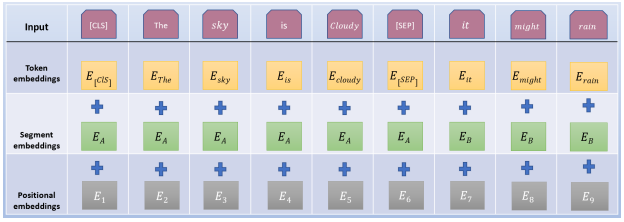


Figure 6: BERT input for NSP task Source:Modified from original version in [4]

The authors of BERT pre-trained the model on unsupervised Wikipedia and Bookcorpus datasets on two language tasks, namely Masked Language Model (MLM) and Next Sentence Prediction (NSP). During MLM training, 15% of the input tokens are replaced with the [MASK] token, and the model is trained to predict the masked word given the context, which consists of the remaining tokens. This approach helps the model to learn the relationship between words within a context. In the NSP task, the model learns relationships and longer-term dependencies between sentences by predicting given two sentences S1, S2 whether S2 is subsequent of S1 or not. In this setting, 50% of input pairs are randomly selected where S2 originally follows S1 in the corpus as positive examples. In contrast, the negative samples are constructed by randomly selecting two sentences

assuming that S2 does not follow S1 in the original corpus [4]. Figure 7 shows the architecture of training and fine-tuning BERT for three different tasks. Simultaneous training, as in BERT, provides better context learning than most earlier language models, trained solely on next word prediction, a direct approach that could limit context learning.

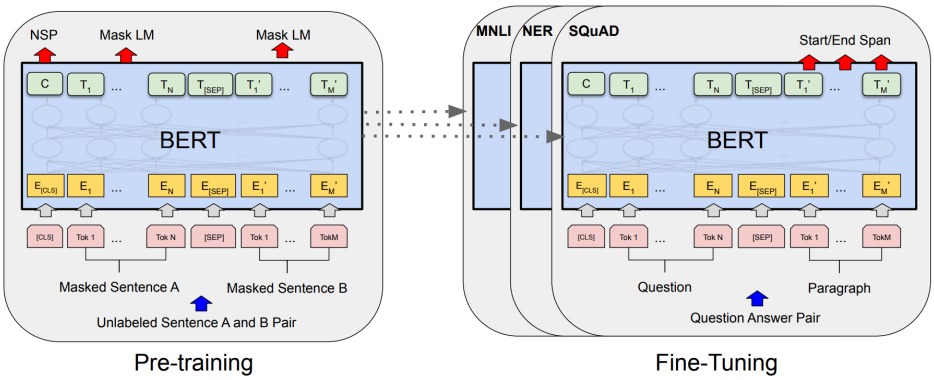


Figure 7: BERT training architecture and fine-tuning. Source: [4]

2.5.3 AraBERT

Despite the success of cross-lingual transfer on various languages, the performance on morphologically complex language is still lagging. To achieve the same success as BERT achieved for the English language, Wissam et al. (2021) proposed AraBERT, a pre-trained BERT for the Arabic language [41]. AraBERT training follows the original training objective using Masked Language Modeling (MLM), which replaces 15% of the input tokens with the [MASK] token 80% of the time and 10% with a random token. The next Sentence Prediction (NSP) task is also employed. The model tries to understand the relationship between two sentences, which can be helpful in tasks like QA.

The model is trained using a large corpus consisting of several open public datasets and manual scraped news articles from different local media across the Arab region. The dataset includes Arabic Wikipedia Dumps, Arabic Corpus El-Khair cite{elkhair201615, containing more than 5 million articles extracted from ten major news sources from different countries, and OSIAN: the Open Source International Arabic News Corpus. The final dataset consists of 70 million, corresponding to 24GB of text.

AraBERT is evaluated on three Arabic language understanding downstream tasks: Sentiment Analysis, Named Entity Recognition, and Question Answering. AraBERT's performance is compared to multi-lingual BERT and other state-of-the-art results for each task. The model outperforms multi-lingual BERT on all sentiment analysis tasks and NER tasks. For QA tasks, AraBERT scores higher on f1-macro and sentence match but slightly lower on exact match. After further examination, the authors explain that most incorrect answers differed by a single or two words without a major impact on the semantics. The large size of the corpus and the vocabulary size can explain the boost in performance in AraBERT in comparison to mBERT.

2.5.4 MARBERT and ARBERT

Recent work by Abdulmajeed et al. attempts to overcome some of the challenges of AraBERT and multi-lingual languages that support Arabic [14]. In their paper, the authors presented two new MLM models focusing on the Arabic language – ArBERT and MARBERT. ArBERT and MARBERT are bidirectional transformer-based models , having the same network architecture as $BERT_{Base}$. ArBERT is trained on Modern Standard Arabic

using a large collection of Arabic datasets corresponding to 61GB of text and a vocabulary of 100K wordpieces. Similar to AraBERT, each training input sequence is generated using whole word masking, where 15% of the N input tokens is selected for replacement, and these tokens are replaced 80% of the time with a [MASK] token, 10% with a random token, and 10% with the original token. Unlike ARBERT, MARBERT is trained on dialect data consisting of 1B tweets sampled from a large in-house dataset of about 6B tweets. Tweets with at least three Arabic words are sampled and used in training. The training dataset consists of 128GB of text with the same vocabulary size of 100K word piece containing MSA and diverse dialects.

Compared to other models such as AraBERT, mBERT, and XLM-Roberta, ARBERT and MARBERT are trained on an enormous collection of Arabic text. mBERT was trained on Arabic Wikipedia dump while XLM-Roberta used CommonCrawel. Although XLM-Roberta contains some dialectal Arabic data, it is unclear how much Arabic data is in the training set. The models were evaluated on different downstream NLU tasks such as sentiment analysis, (2) social meaning (e.g., age and gender, dangerous and hateful speech, emotion, irony, sarcasm), (3) topic classification, (4) dialect identification, and named entity recognition [14].

The authors evaluated the two models on different tasks using XLM-Roberta, mBERT, and Arabert as baselines. For sentiment analysis, the authors evaluated the models on 12 datasets. MARBERT achieves the best results in 10 out of 12 models. ARBERT is best on one dataset while XLM-Roberta is best on two datasets [14]. The results of the sentiment analysis experiments suggest that both models outperform in most cases all other models, MARBERT being superior over ARBERT. MARBERT earns the best performance on 7 of the 8 tasks on social meaning analysis, while ARBERT wins only one (irony@FIRE2019). XLM-Roberta also outperforms AraBERT on 6 of the eight tasks. This is expected from MARBERT as it is primarily trained on dialect-based datasets. As for the topic classification task, ARBERT outperforms other models on 4 out of 5 dataset. MARBERT scores 1% lower compared to to ARBERT AND AraBERT. But the model outperformed both SOTA models and AraBERT in the Dialect Identification task with an average F1 score increase of 5% on all classification [14].

These results reflect the robust and diverse dialectal representation capacity of MARBERT. Although ARBERT is mainly developed for MSA, it outperforms AraBERT, mBERT, and XLM-Roberta with an average of 3% F1 across the different dialect classification tasks. MARBERT also acquires the best results in 3 out of 5 datasets for NER. Overall, MARBERT shows great success over previous models such as mBERT, XLM-Roberta, and AraBERT.

2.5.5 Transformers rankings alternatives

In [5], an alternative methodology for ranking Transformers was proposed. The ColBERT framework uses a novel late interaction paradigm to estimate the relevance between a query q and a document d. With late interaction, q and d are separately encoded in two sets of contextual embeddings, and relevance is determined using cheap and pruning-friendly computations between both sets. The ColBERT framework makes it simple to balance the quality and cost of neural IR, particularly language models like BERT. Figure 8 illustrates the proposed ColBERT architecture. This architecture enables the documents to be processed offline since the document token embeddings are generated independently of the query tokens. The query embeddings interact with every document embedding through a MaxSim operator that computes maximum similarity (e.g., cosine similarity), and the results of these operators are summed across query terms. In addition, it allows ColBERT to retrieve the top-k results directly from a large document collection, resulting in a significant improvement in recall compared to models that only re-rank the output of term-based retrieval. As designed, ColBERT isolates almost all the computations between queries and documents, primarily to enable pre-computing document representations online. ColBERT can serve queries in tens or few hundreds of milliseconds. For instance, when used for re-ranking as in “ColBERT (re-rank)”, it delivers over 170× speedup relative to existing BERT-based models.

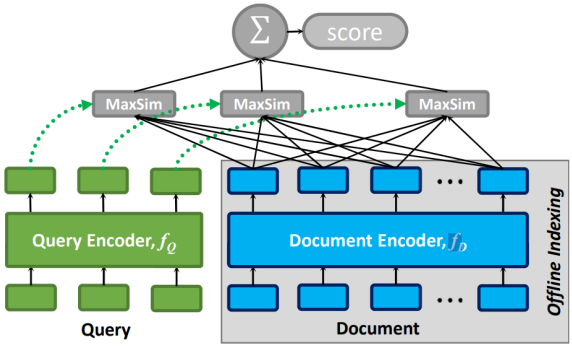


Figure 8: General architecture of ColBERT given a query q and a document d. Source: [5]

2.5.6 BERT in Arabic IR

Recently more studies have adapted transformers for Arabic QA tasks to take advantage of their superior performance across different languages, such as English, Spanish, Indian, and Chinese. The paper [13] proposed SOQAL an open-domain Arabic QA system that uses neural approaches to tackle the reading comprehension task in particular. The system has three modules, a reading comprehension module and a re-ranking module that ranks all possible answers. The retrieval module uses a hierarchal tf-idf, indexing-based model. In this approach, tf-idf is first performed using bi-gram features to retrieve the top-k documents, where k is typically a large number, approximately 1000 and obtain a score for each document denoted DocScore(i). Afterward, a separate tf-idf is constructed for each question and the top-k candidate, only this time using 4-gram features and a much smaller k of around 15. A cosine similarity score is computed between the query and each document retrieved in this phase. The documents are re-ranked using the cosine similarity and the DocScore(i), and the final 15 documents are then passed to the reading comprehension module to extract possible spans of answers. The reading comprehension module is based on multi-lingual BERT that takes the query and passage as input and tries to predict the answer span's start token and end token given by :

$$AnsScore(i) \propto Pstart(i)Pend(i).$$

Finally, to obtain the final answer, we need to combine the scores through a linear combination and choose the candidate that maximizes:

$$(A) - \beta$$

$$arg_{max} i \in [k] \quad \beta\, DocScore(i) + (1-\beta) AnsScore(i).$$

Where

$$\beta \in [0,1]$$

is a hyperparameter chosen through a line search using a development set. The retrieval results on the ARCD dataset outperform the standard tf-idf and Wikipedia API but are about 10% behind the standard google API.

2.6 Transfer Learning

2.6.1 Feature Based Transfer Learning

Features-based transfer learning is achieved by pre-training embedding vectors, which encodes the meaning of the word typically in the form of a real-valued vector as in word2vec [42]. Unlike distributional semantics, which relies on counting the context of a target word w, word2vec instead trains a classifier that predicts how likely word w is in the context of word w [42]. Word2vec uses two approaches to learn the word representation, namely skip-gram and CBOW [42].

Skip-gram treats the target word and a neighboring context word as positive examples while randomly sampling other words in the lexicon as negative examples. A logistic regression-based classifier is trained to distinguish the two cases where regression weights (elements in the vector) are used as the final word embeddings. In the second approach, the CBOW model predicts the word in the middle of a symmetric window based on the sum of the vector representations of the words in the window. Shorter windows with 1-2 words capture more semantically similar words with the same part of speech, while longer windows with 5 or more words capture thematically related words. Other variants, such as the work of Bojanovski et al., in "Enriching Word Embeddings with Subword Information," uses byte-pair encoding for word segmentation, and substring embeddings are essentially learned.

2.6.2 Cross Lingual Transfer Learning

A lot of work is done towards learning language-invariant latent representations in the domain of cross-lingual transfer learning. Languages differ in many ways, including language structure, word structure, vocabulary (lexicon), etc. These differences impose additional challenges in the learning of the underlying model. Data augmentation and sentence level MT-based has been widely used to improve language tasks on languages with few annotated datasets. Although such datasets might exist for a few high-resource languages, such as English, Spanish, and German, most languages lack the labeled data needed to train deep neural networks. It is simply too expensive and time-consuming to collect annotated data for all languages. Until recently, cross-linguistic transfer research has mainly been focused on sequence-to-sequence tasks of POS tagging and parsing due to the lack of high-quality benchmark datasets [43], [44]. As a result of large datasets like MLQA, cross-lingual transfer learning for QA has gained momentum and research in this area has accelerated [45].

2.6.3 QA Cross-Lingual Transfer Learning

When transferring a particular task, it is often unclear which language to use, so the standard strategy is to choose languages based on the experimenter's intuition and some heuristics like selecting a language within the same language family. Having said that, there is no guarantee that all languages within a single language family possess the same linguistic properties. To reduce the burden on the experimenter, the work in [46] considers the task of selecting the optimal transfer language for a given NLP task as a ranking problem. The authors looked at four common NLP tasks: machine translation (MT), entity linking (EL), part-of-speech (POS) tagging and dependency parsing (DEP) and devised a model for each NLP task. To train such a ranker, a suitable dataset is designed by a thorough sweep over a set of training task languages:

$$t1, t2, ..., tn$$

resulting in sets of scores:

$$s1, s2, ..sn$$

A ranking system is then trained using these scores using standard methods for learning to rank. The authors used GBDT [47] an ensemble of Decision Trees DT using Gradient Boost with LamdaRank for the ranker. When considering multiple dataset statistics and language attributes, the learned ranking models recommend much better transfer languages than those predicted by models that consider only a single dataset or language feature. Cross-lingual Transfer learning is beneficial when the two languages share some underlying structure and struggle the most when the source and target languages are too different [48]. In recent years, several approaches have been proposed to leverage training data from an auxiliary language using zero-shot and few-shot cross-linguistic transfer learning. Lately, Generative Adversarial Networks (GANS) based techniques are being explored to learn language-invariant latent representations in many ways. While translation approaches have been proven effective on multi-lingual benchmarks, especially for low-resource languages, the performance heavily depends on the quality of the translator. With fewer linguistic resources, GAN-based approaches are competitive with MT-based methods.

The paper [49] investigated the performance of multi-lingual knowledge graph question answering (KGQA). KGQA aims to answer questions from many languages using a knowledge graph. The authors proposed a framework for inference on multi-lingual questions using adversarial training and unsupervised bilingual lexicon induction (BLI) to convert training questions in the source language into those in the target language as augmented training data. As an alternative to full-supervised machine translation, the authors used BLI for word-level translation. Their strategy has proved effective in dealing with the problem of syntax disorders caused by BLI.

A related study by Microsft examined different strategies to improve multi-lingual transfer learning for multi-lingual QA using translation as a data augmentation technique and adversarial training. The study investigated different ways to improve the multi-lingual embeddings by bringing them closer together in the semantic space so that words with similar meanings (irrespective of the language) are close together in vector space. To tackle these challeneges, the study proposed two methods. First, a novel approach is introduced that incorporates adversarial training (TA) [50]. This strategy aims to train the AT and QA models to make the learned embeddings as language-invariant as possible. In the AT model, discriminator D is trained to classify question representations in different languages correctly. In contrast, the adversarial model is used to push the learned embeddings of other languages together by making the questions appear uniform to the discriminator across all languages. The second method proposed in the paper described a novel Language Arbitration Framework (LAF) to train a multi-lingual QA model. This scenario ensures that the same question generates the same answer across different languages. This study evaluated the models using MLQA and TyDi QA datasets. Both AT training and LAF strategy improve the f1 score when tested on different languages, including Arabic, and outperforms the previous zero-shot baseline.

2.6.4 Multi-Task Transfer Learning

Representation of text, in general, is a fundamental problem in natural language understanding (NLU). A growing interest has emerged in applying MTL to representation learning using deep neural networks (DNNs). Supervised learning in DNN requires a large amount of annotated data for a given task. The ML community has studied multi-task learning since the mid-90s. Lorien Pratt and Sebastian Thrun define multi-tasking as an inductive transfer mechanism whose principal goal is to improve generalization performance [51]. It can be achieved by providing a more substantial inductive bias than it would otherwise have without the extra knowledge. Multi-task algorithms exploit the structure and similarities across different learning problems and are based on the idea that tasks are related through a common low dimensional representation, which is learned jointly with the tasks' parameters [52].

MTL provides an efficient way to leverage domain-specific information from one task across different domains, improving generalization. Furthermore, MTL adds a regularization effect by reducing overfitting to a specific task, thus making the learned representation universal across tasks. Unlike MTL, pretraining language models are more effective for learning universal language representations when working with large amounts of unlabeled data. Figure 10 shows an example architecture for multitask when the output variable y has the same semantics for all tasks while the input variable x has a different distribution [1]. In this setting, the upper levels from $h^{(shared)}$ on-wards are shared while the lower levels are task-specific. The model tries to convert each task-specific input into a generic representation.

2.6.5 Research in MTL

A recent study by Microsoft proposes a new Multi-Task Deep Neural Network that combines both MTL and language model pretraining technologies [53]. The authors argue that this improves text representation learning, improving the performance on various NLU tasks. The architecture of the MT-DNN model consists of

shared and task-specific layers. All tasks share the lower layers, including the transformer layer, whereas the top layers represent the outputs for specific tasks. The transformer layer captures the contextual information for each word via self-attention and produces a common representation that simultaneously simplifies the underlying specific tasks. Lastly, task-specific layers generate representations for each task used for classification, similarity scoring, or relevance ranking operations. The task-specific layers use the common representation for its downstream task.

MT-DNN achieves new state-of-the-art results on various NLU tasks, including SNLI, SciTail, and almost all GLUE tasks [53]. In another study, the knowledge distillation method introduced by Hinton et al. was further extended to improve MT-DNN for learning text representation [54], [55]. In this setting, an ensemble of MT-DNN is first trained for tasks with an available dataset and acts as a teacher model. In an offline manner, the ensemble is utilized to produce a set of soft targets for each x in the training dataset. Then a single MT-DNN (student) is trained using the generated soft targets and the correct targets via multi-task learning. In this manner, the generalization capabilities of the teacher are effectively transferred to the student. Using the same training data as those used to train the teachers, the distilled MT-DNN outperforms the vanilla MT-DNN that was trained in a traditional manner [55].

Although both vanilla MT-DNN and distilled MT-DNN can significantly improve model performance across many NLU tasks, serving an ensemble of large DNNs and training multiple DNNS in a teacher/student manner can be very challenging and resource extensive.

2.6.6 Adapters

Motivation Transfer learning techniques, including feature-based transfer learning and fine-tuning large pre-trained models, have been effective in many complex NLP tasks such as machine translation and QA. Nevertheless, fine-tuning transfer learning (eg. BERT, GPT2, XLM) often yields better performance than feature-based [4], [56], [57], [58]. The study in [58] has empirically analyzed fine-tuning, and feature extraction approaches across diverse datasets, suggesting that fine-tuning language models significantly outperform feature extraction methods. Models such as BERT and GPT2 apply parameter transfer learning where the target tasks can benefit from the knowledge previously encoded in the parameter space during the training phase of the source task. These models are generally characterized by their number of parameters and the depth of the underlying deep

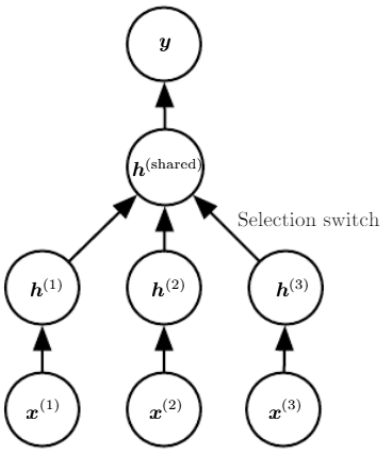


Figure 9: Example architecture for multitask or transfer learning when the output variable y has the same semantics for all tasks while the input variable x has a different meaning. Source: [1]

neural networks.

Furthermore, increasing these factors improves the model's capacity and performance, as seen in GPT3 compared to its predecessor GPT2 [59]. In light of their success, one would expect more models with billions or trillions of parameters to be published. While fine-tuning proves to be successful, it comes at a considerable training cost. In addition to being expensive, slow, and time-consuming, storing and sharing large trained models hinders advances towards more general and versatile NLP methods.

Fine-tuning requires training an entire model for a new set of weights for every task, making it inefficient as a parameter sharing technique. As an alternative to full fine-tuning, [60] first introduced adapters as a lightweight fine-tuning strategy that can achieve comparable performance level to full fine-tuning [61] on many tasks. The use of adapters, i.e., learned bottleneck layers inserted within each layer of a pre-trained model, alleviates this issue by avoiding the need to fine-tune the entire model. By sharing the lower layer of a network between tasks, fine-tuning can be more parameter efficient than feature-based transfer. Using this approach, we can efficiently share parameters between tasks by training multiple adapters, some of which are task- and language-specific, which can be easily exchanged, combined, and resed after training.

Pfeiffer et al. have shown promising results in multi-task and cross-lingual transfer learning with adapters based fine-tuning [62]. Adapters add a new set of weights at every layer of the transformer and are trained during the fine-tuning phase while keeping the original pre-trained parameters fixed. Freezing the original parameters significantly reduces the number of trainable weights and hence minimizes the model training time and yields a high degree of parameter sharing. Nevertheless, sharing and reusing adapters is not straightforward, as adapters are rarely released individually, differ in subtle yet significant ways, and are model- and task-specific [8].

2.6.7 AdapterHub

The paper "AdapterHub: A Framework for Adapting Transformers" introduces an easy-to-use framework that facilitates the dynamic integration of pre-trained adapters for different tasks and languages [8]. Based on HuggingFace Transformers, the framework supports quick and easy adaptation of pre-trained models (e.g. BERT, ROBERTa, XLMR) across tasks and languages [63]. They are easily inserted within every transformer layer. In the AdapterHub framework, the pre-trained weights are fixed, and only the adapter weights and the prediction head are trained, forcing the model towards a compatible representation. Scalability, modularity, and composition are just a few of the benefits of adapters over fine-tuning a model. Below are a number of useful use-cases as discussed by the authors:

1. Task-specific Layer-wise Representation Learning: provides a lightweight alternative to full fine-tuning with comparable performance.

2. Small, Scalable, Shareable: 99% of the parameters for each task are shared across all models, reducing the size and increasing scalability and ease of share.

3. Modularity of Representations: Adapters learn to encode information of a task within designated parameters. The encapsulated placement of adapters between layers, forces the adapters to learn an output representation that is compatible with the next layer of the transformer model.

4. Non-Interfering Composition of Information: Adapters allow us to leverage information from specific tasks, domains, or languages that are more relevant to a particular application rather than from general pretrained counterparts.

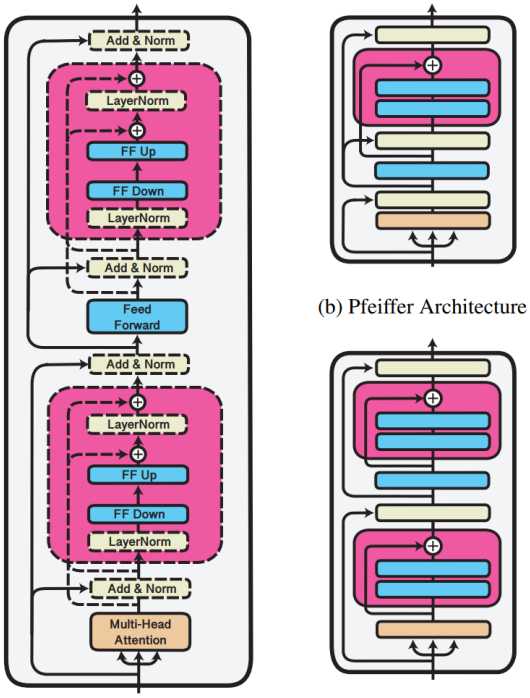


Figure 10: Dynamic customization possibilities where dashed lines in (a) show the current configuration options. These options include the placements of new weights (including down and up projections as well as new LayerNorms), residual connections, bottleneck sizes as well as activation functions.The resulting configurations for the architecture proposed by Pfeiffer et al.[6] and Houlsby et al. [7] are illustrated in (b) and (c) Source: [8]

3 Methods

This chapter explains our experiments' methodologies for implementing the passage retrieval and reading comprehension modules. The study evaluates BERT-based models' performance on two tasks: passage re-ranking and answer extraction from a span of text. As part of these tasks, several transformer-based models are constructed and evaluated using fine-tuning and parameter-efficient transfer learning techniques. In particular, the study compares adapter-based transformers and non-adapter-based in re-ranking and MRC settings. The first section describes the datasets used to train and evaluate the proposed models for each task. The following section explores the tasks formulation and training environment in more detail.

3.1 Datasets

3.1.1 MS MARCO Dataset

MS MARCO is a large-scale machine reading comprehension that contains 1,010,916 anonymized questions collected from Bings' search logs, and 8,841,823 passages from different websites [20]. Each question has a human-generated answer, and 182,669 human re-written generated answers. The passages are extracted from Bing's web documents, containing the necessary information to answer the query. To summarize, the published dataset consists of the following six components:

1. Questions: anonymized questions collected from Bing's search log.
2. Passages: at least ten relevant passages are collected for each query.
3. Answers: each question has zero or more manually written answers by human editors.
4. Well-formed Answers: re-written and well-reviewed answers by human editors.
5. Document: consists of the web document URL, title, and body text
6. Question type: defining the question type which could be NUMERIC, ENTITY, LOCATION, PERSON, DESCRIPTION [20].

The fact that the queries are user-generated sets MS MARCO apart from other well-known datasets for machine learning and makes it attractive for benchmarking MRC models. The authors proposed three tasks for the MS MARCO dataset:

1. Binary prediction task to identify if a passage is relevant to a given query.
2. Generate well-formed answer based on the query and context given.
3. Given a query and a set of passages rank the passages based on relevancy.

The passage ranking task was formulated based on the passages and questions in the Question Answering (QnA) Dataset. The objective is to score a set of passages in a corpus of 8.8 million passages based on relevancy. Passages containing answers to the Question Answering dataset's queries are labeled relevant.
The original dataset is in English, and machine-translated versions are available for other languages. The work in [64] provides a neural translation version of the dataset in the Arabic language, which is primarily used for training our passage ranker module. The dataset contains 8820392 passages and queries. Sample training records are shown in Table 2 .

Query	حيث جزيرة وايت مارش
Relevant passage	زيرة بي هي جزيرة تقع في ولاية كارولاينا الشمالية. وبسبب الطبيعة المتغيرة لنظام الحزر الحاجز الذي تشكل جزيرة بيا جزءا منه، والطريقة التي يتم بها فتح
Irrelevant passage	ما في ذلك فيري إلى جزيرة ليبرتي وجزيرة إليس
Irrelevant passage	شروق الشمس ٦ أكتوبر ٢٠٠٨ جزيرة هاربور، جزر البهاما، من الجزء الشمالي من الجزيرة. تدريب الفرقة الأفريقية في نادي
Irrelevant passage	ميريت ايلاند تقع مدينة ميريت غير المدمجة في جزيرة فلوريدا الكبرى، التي تعمل كموطن لمركز كينيدي الفضائي التابع لناسا، ولا تزال تمثل جاذبية سياحية كبرى

Table 2: Sample records from Arabic MS MARCO dataset

3.1.2 Stanford Question Answering Dataset (SQuAD)

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset consisting of crowdsourced question-answer pairs from Wikipedia articles [65]. To build SQuAD, the creators examined 536 of the top 10,000 Wikipedia articles covering a wide range of topics. Across all sampled articles, they extracted 23,215 unique paragraphs, excluding smaller ones. 80% of the articles were used to build the training set, 10% for the development set, and 10% for the testing set [65]. For each paragraph, the annotators were tasked to generate five questions about the paragraph's content using their own words. In addition, they had to answer five questions by highlighting the appropriate span of text from the given content [65]. To train our MRC module, we used a machine-translated version of SQuAD v1.0 with 48,344 questions on 10,364 passages [13].

3.1.3 ARCD Dataset

ARCD is a crowdsourced Arabic Reading Comprehension Dataset consisting of 1,395 human-generated questions [13]. A similar approach to the data collection and annotation approach described in [65], was applied in ARCD data collection process [65]. For each task, the crowd workers were presented with the top three paragraphs of the five articles from the Arabic Wikipedia. The task is to compose three question-answer pairs for each paragraph in MSA, where the answers are the exact span of text from the provided content. Following the work in [13], and [14], we evaluate the performance of our modules on ARCD datasets consisting of real human queries.

3.1.4 TyDi QA Dataset

Making progress in multi-lingual modeling requires challenging, reliable evaluations. The TyDi QA dataset includes 204K question-answer pairs across 11 typologically diverse languages varying in their linguistic features. With such diversity, models performing well on TYDI QA are expected to generalize well across different languages [66]. It is developed with two purposes in mind:

1. improves the quality of question answering systems of the world's top 100 languages in the world.
2. encourages the development of models that function well across multiple linguistic phenomena.

To enable a realistic information-seeking experience, the data annotators are asked to write real questions that they are interested to learn about. The queries are collected in each language directly without machine translation for better quality. Table 3 shows sample queries and articles from the TyDi QA dataset.

3.2 Tasks

3.2.1 BERT for Passage Ranking

The goal of the passage re-ranking task is to rank a set of documents based on the relevancy to a given query and is treated as a binary classification problem. The most straightforward way to build a passage ranking system

using transformers is to convert the problem into a binary classification task with two relevant and non-relevant classes. The passages are ranked based on their probabilities of belonging to the desired class. It is a very straightforward approach that follows the recommendations from the original BERT paper as to how to fine-tune with BERT. Using training data, one can fine-tune the BERT model to learn a representation of documents and queries so that relevant documents are more similar in that representation than irrelevant documents.BERT takes as input a sequence of tokens (more specifically, input vector representations derived from those tokens). It outputs a sequence of contextual embeddings, which provide context-dependent representations of the input tokens. This stands in contrast to context-independent representations, including many of the widely adopted techniques that came before, such as word2vec [42] or GloVe [67]. The query and passage are concatenated for each training example and separated by the special token [SEP]. The passages are truncated such that the combined query, passage, and separator tokens have a maximum length of 512 tokens. The [CLS] vector representation is used as input for the binary classification layer to obtain the final score.

The proposed models are trained on a subset of the Arabic version of MS MARCO consisting of 4 million question-answer pairs and passages in total. For each record, the output is 0 if it is not relevant and one if it is relevant. 80% of the dataset is used for training, while 20% is used for validation and parameter tuning. The passages are indexed for faster retrieval using Elastic Search for evaluation purposes. The top hits (max-100) are fetched using the BM25 algorithm and passed to the re-ranking module for each query in the set. After re-ranking, the MRR@100 is calculated on the sorted set, and the AVP@100 is computed as the average MRR@100 over all queries in the evaluation. MRR@10 and AVP@10 are computed similarly.

3.2.2 BERT for MRC

The main goal of a typical MRC task is to require a machine to identify an answer to a query within a set of passages [68]. It is commonly modeled by span labeling, where the model is trained to predict the answer span by identifying the start word index and the end word index of the answer span. The query and passage are concatenated using a [SEP] token as shown in Figure 11. For a given question q of n tokens

$$q_1, ..., q_n$$

and a passage p of m tokens

$$p_1, ..., p_m$$

the goal is to compute the probability

$$P(a|q,p)$$

that each possible span a is the answer [9]. For each token w_i in passage p, the algorithm computes two probabilities for a possible answer candidate: $pstart(w_i)$, which is the probability that w_i is the start token of the answer, and $pend(w_i)$ the probability that w_i is the end of the answer span [9]. This probability can be estimated as

$$P(answer|q,p) = P_{start}(w_{start}|,q,p).P_{end}(w_{end}|q,p)$$

, where w_{start} and w_{end} are tokens indicating the start and end words within the passage p.

A linear layer is added to predict the start and end positions of the answer span together with two new embeddings, span-start embedding S and a span-end embedding E that will be learned during fine-tuning. To get a span-start probability for each output token i, the dot product between S and $p_i^{'}$ is computed for each token, and then a softmax operator is used to normalize overall tokens $p_i^{'}$ in the passage:

$$p_{start_i} = \frac{exp(S.p_i^{'})}{\sum_j exp(S.p_j^{'})}$$

The same equation is applied for computing the end word probabilities :

$$p_{end_i} = \frac{exp(E.p_i^{'})}{\sum_j exp(E.p_j^{'})}$$

The training loss is computed as the negative sum of the log-likelihoods of the correct start and end positions for each example:

$$L = logP_{start_i} - logP_{end_i}$$

The model returns the span that maximizes the probability:

$$P(answer|q,p) = P_{start}(w_{start}|,q,p).P_{end}(w_{end}|q,p)$$

as described in section 2.3.2.

3.3 Baselines

1. **Passage Re-Ranking:** The performance of the BERT-based passage ranking models is evaluated against the BM25 baseline ranker. The re-ranking is performed on the top 100 and top 10 retrieved documents and is evaluated using the average precision and reciprocal re-rank metrics described in the next section.

2. **Language Model Baselines:** multi-lingual BERT and Roberta-XLM are used as baselines for evaluating different SOTA mono-lingual BERT-based models on passage ranking and MRC tasks. While such multi-lingual LMs are often outperformed by mono-lingual models pre-trained on language-specific datasets, they form solid baselines for evaluating language-specific models [69], [70].

3.4 Preprocessing

Light preprocessing is performed on the raw data before passing the text to the wordpiece tokenizer. This helps to maintain a faithful representation of the naturally occurring text. As in [14] Diacritics, URLs, and HASH Tags are removed before tokenization

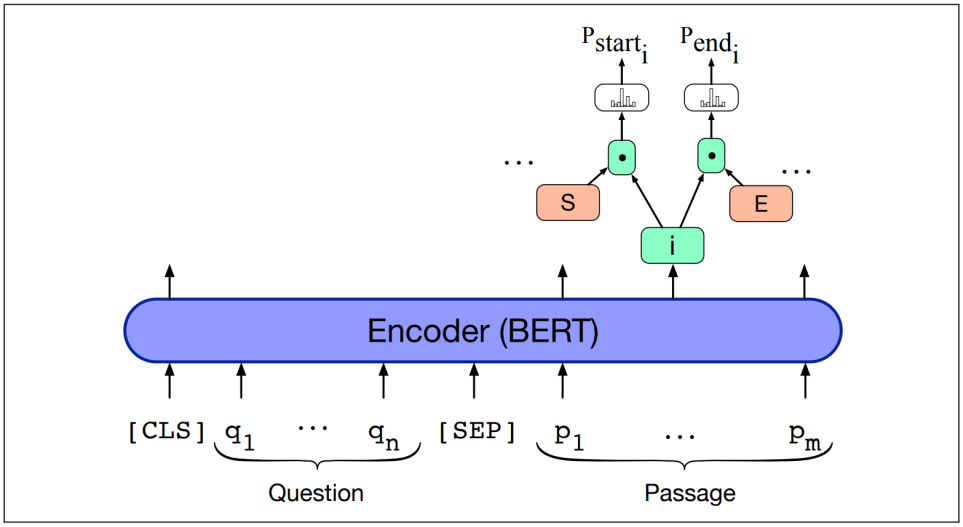


Figure 11: BERT encoder model for span-based question answering. Source: [9]

Query	Answer
أين تقع مدينة أوديسا؟	جمهورية أوكرانيا
من هو مخترع المحرك البخاري؟	جيمس واط
في أي عام توفي إسحاق رابين؟	٤ نوفمبر ١٩٩٥
اين توجد منطقة الخفجي؟	شمال شرق المملكة العربية السعودية
ما هو عدد آيات سورة الكهف ؟	١١٠

Table 3: Question Answer Pairs from TyDi QA dataset

4 Experimental Setup and Results

This chapter describes the tools and technologies used in the experiments and ends with the results of the experiments. The first four subsections introduce the different tools used to set up passage re-ranking tasks, proposed models, experimental configurations, and evaluation metrics. Two experiments are carried out for each task, one using fine-tuning and the other using adaptive tuning. The results of the four experiments are provided at the end of the section.

4.1 Tools and Technologies

Python Elasticsearch Client version 7.6.2 is employed to retrieve the relevant passages using the BM25 algorithm. Python Elasticsearch Client is the Official low-level client for Elasticsearch designed as a wrapper around Elasticsearch's REST API [71]. The library is configured to use BM25 as a default retrieval mechanism. The article collection consisting of ARCD and TyDi QA articles are first indexed for faster retrieval. During the re-ranking phase, the top-k articles are first retrieved using the Elasticsearch Client and then re-ranked using the proposed models for evaluation.

4.2 Proposed Models

We present different SOTA BERT-based models to evaluate adapter-tuning versus fine-tuning. For each setting and task, the model is trained separately and evaluated. Table 4 shows the proposed models and settings. BERT-based models can be divided into two groups: multi-lingual models and monolingual models. Multi-lingual models form the basis of our baseline as described in the previous section and are critical in evaluating monolingual models for low resource languages such as Arabic.

Model	Number of Languages	Number of trainable parameters
mBERT-Base (cased)	104	177264386
XLM-Roberta (base)	100	124056578
AraBERT	1	134604290
ARBERT	1	162252290
MARBERT	1	162252290
mBERT-Base (cased) + Adapter	104	896066
XLM-Roberta (base) + Adapter	100	896066
AraBERT + Adapter	1	896066
ARBERT + Adapter	1	896066
MARBERT + Adapter	1	896066

Table 4: Proposed models and associated number of languages and number of trainable parameters

4.3 Experimental Configurations and Hyperparameter Optimization

To fine-tune BERT-based models apart of XLM-Roberta, we search different learning rates as suggested in [4]. In our case, Stochastic Gradient Descent with a learning rate of 1e-4 provides the best results for mBert while 1e-5 provides the best outcome for ARBERT, MARBERT, and AraBERT. For the XLM-Roberta model,

the initial learning rate of 1e-3 provided the best development results. All models are trained using a scheduler with a decay rate of 0.9. For adapter-tuning, we used AutoModelWithHeads as a base model instead of AutoModelForQuestionAndAnswering. For each task, a task adapter is applied using Pfeiffer architecture [8] and trained using learning of 1e-4 while freezing the model's original parameter. For passage re-ranking, the maximum sequence length is set to 128 due to memory limitation during training and evaluation. A sequence length of 512 is used across all models for the reading comprehension task.

Model	mBERT	XLM-Roberta	ARBERT	MARBERT	AraBERT
Vocabulary size	110k	50k	100K	100K	64k
Optimizer	SGD	Adam	SGD	SGD	SGD
Learning Rate	$1e^{-4}$	$3e^{-3}$	$1e^{-5}$	$1e^{-5}$	$1e^{-5}$
hidden dropout prob	0.1	0.1	0.1	0.1	0.1
epoch	1	1	1	1	1
Train batch size	64	64	64	64	64
max seq length	128	128	128	128	128
word embedding size	768	768	768	768	768

Table 5: Hyperparameter Optimization used in fine-tuning the proposed models for passage re-ranking

Model	mBERT	XLM-Roberta	ARBERT	MARBERT	AraBERT
Vocabulary size	110k	50k	100K	100K	64k
Optimizer	Adam	Adam	Adam	Adam	Adam
Learning Rate	$3e^{-5}$	$3e^{-5}$	$1e^{-5}$	$1e^{-5}$	$3e^{-5}$
hidden dropout prob	0.1	0.1	0.1	0.1	0.1
epoch	1	1	1	1	1
Train batch size	64	64	64	64	64
max seq length	512	512	512	512	512
word embedding size	768	768	768	768	768

Table 6: Hyperparameter Optimization used in fine-tuning the proposed models for MRC task

4.4 Evaluation Metrics

This subsection describes the evaluation metrics used for the task of passage re-rank and MRC. We use the mean reciprocal mean and average precision for passage re-rank. F1 score and exact match metrics are used for the task of MRC.

Model	mBERT	XLM-Roberta	ARBERT	MARBERT	AraBERT
Vocabulary size	110k	50k	100K	100K	64k
Optimizer	Adam	Adam	Adam	Adam	Adam
Learning Rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
hidden dropout prob	0.1	0.1	0.1	0.1	0.1
epoch	1	1	1	1	1
Train batch size	64	64	64	64	64
max seq length (re-ranking)	128	128	128	128	128
max seq length (MRC)	512	512	512	512	512
word embedding size	768	768	768	768	768

Table 7: Hyperparameter Optimization used in adapter-tuning the proposed models for re-ranking task and machine reading comprehension

4.4.1 Mean Reciprocal Rank

The Mean Reciprocal Rank MRR is a commonly used metric to evaluate retrieval systems [72]. It is the official re-ranking metric for the MS MARCO dataset. The Reciprocal Rank (RR) measure is the reciprocal rank of the first relevant document. For a given query, the RR is

$$\frac{1}{rank}$$

, where rank is the position of the highest-relevant document in the retrieved N documents. RR is 1, if the relevant document is placed first in the list

$$RR = \frac{1}{1}$$

and is 0.5 if it is ranked second

$$RR = \frac{1}{2}$$

For multiple queries, the MRR metric is used, which averages the RR across all queries as shown in Equation 7. MRR is suitable for systems where a single relevant document is expected for each query. In such scenarios, other relevant documents scored in positions 2, 3, 4, etc., don't affect the overall performance.

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{7}$$

4.4.2 Mean Average Precision

In IR systems such as search engines, where the user is interested in finding many relevant documents, the Mean Average Precision is normally used instead of the MRR. Given a set of d relevant documents $d_1, d2, .., dk$, the Average Precision@D for each document is computed as in

$$AveragePrecision = \frac{\sum_{i=1}^{k} Precision@d_i}{K}$$

and the Mean Average Precision is then the mean over all queries and ranks. Algorithm 1 describes the logic used for evaluation.

Algorithm 1 Evaluation loop for passage re-ranking task

```
mrrlist ← [] empty list
avplist ← [] empty list
for each query q, answer a in evaluation set do
    scores ← [] empty list
    passages ← [] empty list
    results ← elasticsearch(q, k = 100)
    for each passage p in results do
        encodedinput ← tokenize(q, p)
        scores ← ranker(encodedinput) append score
        passages ← p append passage
    sortedindexes ← sort(scores) descending
    sortedpassage ← sort(passages) using sorted idx
    binaryresults ← [] empty list
    for each passage p in sorted passages do
        if Answer in passage then
            binaryresult ← 1 append
        else binaryresult ← 0
    mrrlist ← compute mrr(binary results) append to list
    avplist ← computer avp(binary results) append to list
mrr@100 ← avg(mrrlist)
avp@100 ← avg(avplist)
```

4.4.3 F1 score

In many classification problems, precision, recall, and F1 are used to assess model quality. Precision is given by

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{8}$$

Precision is a good measure to determine the percentage of positive observations correctly predicted from the total positive predictions. The recall is the ratio of the positive observation correctly predicted to all observations in the positive class and is given by

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{9}$$

F1 score also referred as the harmonic mean, is the weighted average of the Precision and Recall. Mathematically it is defined as

$$F1 = \frac{2 * Recall * Precision}{(Recall + Precision)} \tag{10}$$

In the context of QA, f1 is computed over the individual predicted words against those in the True Answer. The F1 score is determined by the number of words shared between the prediction and the truth.

4.4.4 Exact Match

Exact Match calculates the percentage of predictions that exactly match the ground truth answer. EM is one if the prediction characters exactly match the gold truth; otherwise, it is 0.

4.5 Results

This section presents the results of the experiments as described in Chapter 4. The proposed five models are evaluated using fine-tuning transfer learning and adapter tuning for each task. As indicated in chapter 4, different experiments are conducted to handle longer texts. We compared the results of splitting the text into smaller chunks and taking the average score, the sum of scores, and the maximum score of the chunks. The results reported here are based on truncating overflowing text, which empirically performs better than the previous attempts. The results of the passage re-ranking task and the reading comprehension task are described in the following subsections.

4.5.1 Experiment 1: Fine-Tuning on Passage Re-Rank Task

For passage re-ranking, the models are evaluated using four metrics: MRR@10, MRR@100, AVP@10, and AVP@100. Average precision is a better performance indicator for queries with multiple relevant passages. Table 8 shows the results of fine-tuning the proposed models on ARCD dataset. In this setting, ARABERT outperforms all models achieving the highest score on all metrics followed by MARBERT and AraBERT. In general, monolingual models are the only ones that outperform the baseline in this setting. The results on Tydi QA dataset are best described in Table 9. Similarly, all monolingual models outperform the baseline, but unlike the previous results, mBERT, a cross-lingual model, performs slightly better than the BM25 baseline.

Metric / Model	BM25	mBERT	XLM-Roberta	ARBERT	MARBERT	AraBERT
MRR@10	0.738	0.671	0.535	**0.793**	**0.756**	**0.753**
MRR@100	0.740	0.675	0.541	**0.768**	**0.759**	**0.754**
AVP@10	0.733	0.668	0.535	**0.787**	**0.752**	**0.745**
AVP@100	0.714	0.649	0.525	**0.768**	**0.735**	**0.725**

Table 8: Results on ARCD dataset, evaluated on the top 1000 retrieved passages.

Metric / Model	BM25	mBERT	XLM-Roberta	ARBERT	MARBERT	AraBERT
MRR@10	0.829	**0.835**	0.789	**0.888**	**0.852**	**0.847**
MRR@100	0.831	**0.836**	0.791	**0.888**	**0.853**	**0.849**
AVP@10	0.821	**0.824**	0.779	**0.880**	**0.842**	**0.839**
AVP@100	0.782	**0.789**	0.743	**0.838**	**0.806**	**0.803**

Table 9: Results on TyDi QA dataset, evaluated on the top 100 retrieved passages.

4.5.2 Experiment 2: Adapter-Tuning on Passage Re-Rank Task

Now that we have identified the performance of each model using the fine-tuning technique let us turn to parameter efficient transfer learning-based training. In the adapter tuning setting, a task adapter is applied to each model and is fine-tuned on the MS MARCO dataset while freezing the original parameters. The results of adapter-tuning on ARCD and TyDi QA datasets are described in Table 10 and Table 11 respectively. For the ARCD dataset, monolingual models again outperform the baseline while cross-lingual models fail to meet the

baseline BM25 benchmark.

To highlight the impact of parameter efficient transfer learning on performance, we present the difference in performance after applying adapter tuning compared to fine-tuning on the ARCD and TyDi QA datasets. Table 12 shows the difference in performance after applying adapter tuning on ARCD data-set. Table 13 shows the difference when applied on TyDi QA dataset. There is a very slight drop in performance on most models, except for XLM-Roberta, which slightly improves its average precision performance on the ARCD dataset. Similar results are also obtained on TyDi QA as shown in Table 13. Most models also drop in performance, but MARBERT improves on MRR@10, MRR@100, and AVP@100. The overall impact of adapter tuning is very minimal, as can be concluded by the results. On ARCD, mBERT is most affected, while XML-Roberta is most affected when tested on TyDi QA. Figure 12 and 13 shows the performance of fine-tuning and adapter tuning of mBERT on ARCD and TyDi QA respectively. In the case of mBERT the drop in performance is not negligible, as shown in Figure 12. XLM-Roberta achieves the lowest score on ARCD as can be shown in Figure 20 and Figure 21. ARBERT on the other hand scores the highest score on all metrics as can be shown in Figure 18 for ARCD and Figure 15 for TyDi QA dataset. Adapter-tuning has very little impact on MARBERT, as shown in Figure 16 and 17 for ARCD and TyDi QA, respectively.

Model	MRR@10	MRR@100	AVP@10	AVP@100
Baseline BM25	0.738	0.740	0.733	0.714
mBERT (Adapter-Tuning)	0.634	0.638	0.629	0.619
XLM-Roberta (Adapter-Tuning)	0.518	0.526	0.515	0.505
ARBERT (Adapter-Tuning)	**0.781**	**0.783**	**0.777**	**0.761**
MARBERT (Adapter-Tuning)	**0.759**	**0.762**	**0.753**	**0.737**
AraBERT (Adapter-Tuning)	**0.754**	**0.756**	**0.750**	**0.729**

Table 10: Adapter-tuning Results on ARCD dataset, evaluated on the top 1000 retrieved passages.

Model	MRR@10	MRR@100	AVP@10	AVP@100
Baseline BM25	0.829	0.831	0.821	0.782
mBERT (Adapter-Tuning)	0.812	0.814	0.802	0.766
XLM-Roberta (Adapter-Tuning)	0.789	0.791	0.779	0.743
ARBERT (Adapter-Tuning)	**0.875**	**0.876**	**0.868**	**0.825**
MARBERT (Adapter-Tuning)	**0.851**	**0.852**	**0.843**	**0.802**
AraBERT (Adapter-Tuning)	**0.831**	**0.833**	**0.823**	**0.785**

Table 11: Adapter-tuning Results on TyDi QA dataset, evaluated on the top 100 retrieved passages.

Model	MRR@10	MRR@100	AVP@10	AVP@100
mBERT (Adapter-Tuning)	-0.0373	-0.0363	-0.03905	-0.0299
XLM-Roberta (Adapter-Tuning)	-0.0169	-0.015	-0.0193	-0.0205
ARBERT (Adapter-Tuning)	-0.01181	-0.0122	-0.0098	0.00699
MARBERT (Adapter-Tuning)	-0.0026	-0.0029	-0.0014	-0.00215
AraBERT (Adapter-Tuning)	0.0013	0.0023	0.0051	0.0043

Table 12: Adapter-tuning impact on MRR@10, MRR@100, AVP@10 and AVP@100 compared to Fine-Tuning of the entire model on ARCD

Model	MRR@10	MRR@100	AVP@10	AVP@100
mBERT (Adapter-Tuning)	-0.02210	-0.02170	-0.02203	-0.02291
XLM-Roberta (Adapter-Tuning)	-0.0560	-0.0561	-0.0540	-0.0430
ARBERT (Adapter-Tuning)	-0.01307	-0.0129	-0.01203	-0.01337
MARBERT (Adapter-Tuning)	0.0008	0.0008	-0.0006	0.0036
AraBERT (Adapter-Tuning)	-0.0164	-0.0163	-0.0156	-0.0177

Table 13: Adapter-tuning impact on MRR@10, MRR@100, AVP@10 and AVP@100 compared to Fine-Tuning of the entire model on TyDi QA

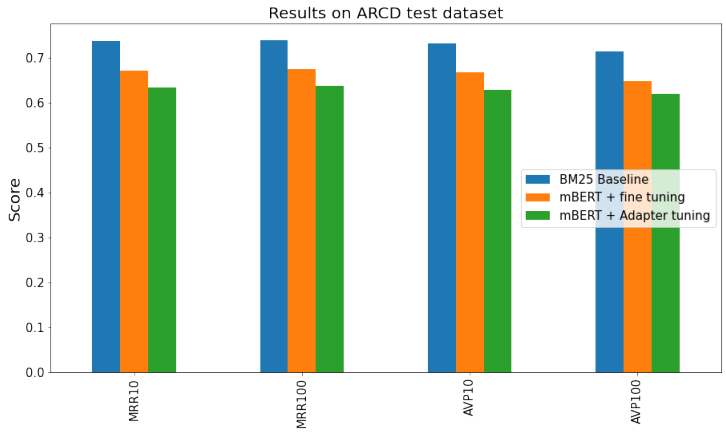


Figure 12: results of fine-tuning mBERT and adapter tuning of mBERT evaluated on ARCD test dataset

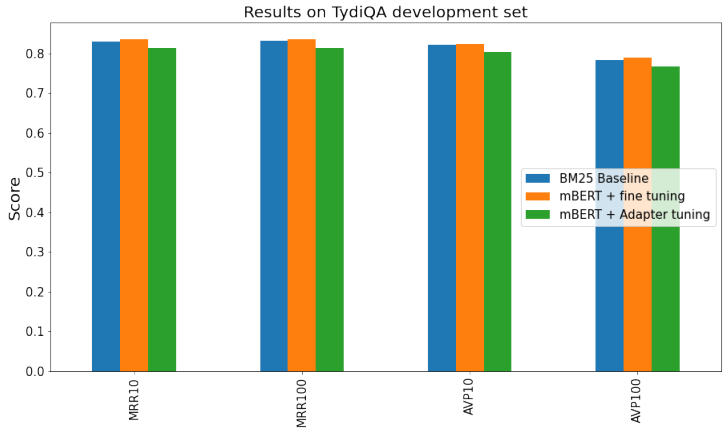


Figure 13: results of fine-tuning mBERT and adapter tuning of mBERT evaluated on TyDi QA test dataset

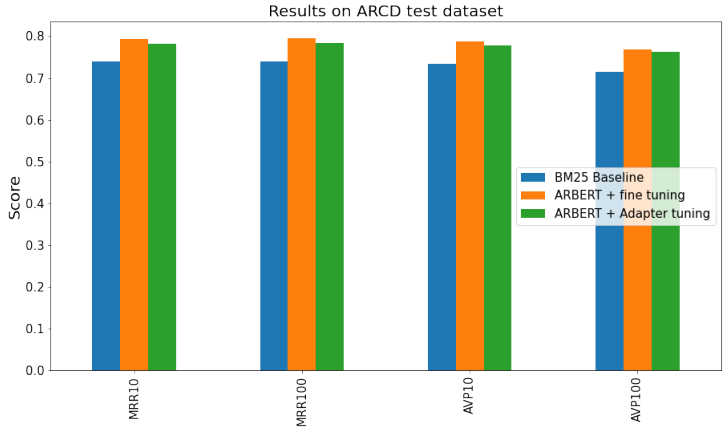


Figure 14: results of fine-tuning ARBERT and adapter tuning of ARBERT evaluated on ARCD test dataset

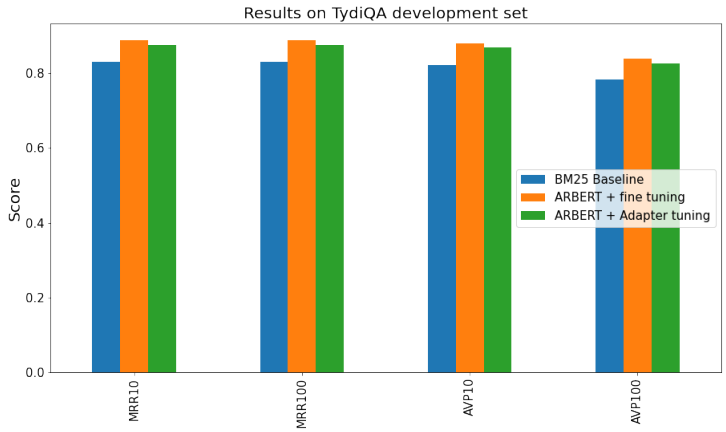


Figure 15: results of fine-tuning ARBERT and adapter tuning of ARBERT evaluated on TyDi QA test dataset

Figure 16: results of fine-tuning MARBERT and adapter tuning of MARBERT evaluated on ARCD test dataset

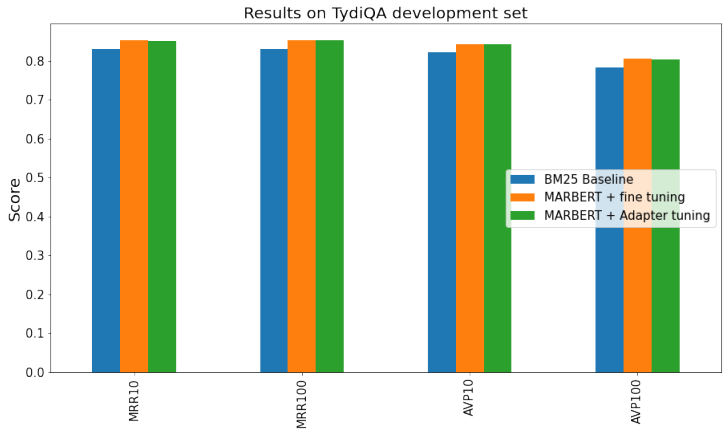


Figure 17: results of fine-tuning MARBERT and adapter tuning of MARBERT evaluated on TyDi QA test dataset

Figure 18: results of fine-tuning AraBERT and adapter tuning of AraBERT evaluated on ARCD test dataset

Figure 19: results of fine-tuning AraBERT and adapter tuning of AraBERT evaluated on TyDi QA test dataset

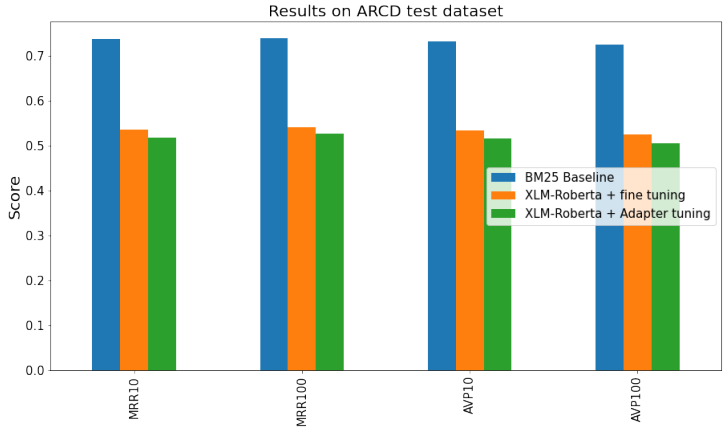


Figure 20: results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta evaluated on ARCD test dataset

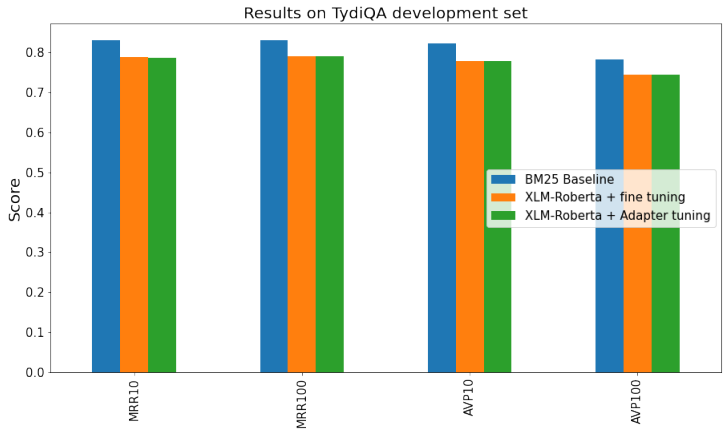


Figure 21: results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta evaluated on TyDi QA test dataset

4.5.3 Experiment 3: Fine-Tuning on Reading Comprehension Task

The models are trained on the Arabic machine translation of the SQuAD 1.0 dataset for the reading comprehension experiments and are evaluated on ARCD and TyDi QA. For evaluation, F1 and EM are presented for each experiment. Fine-tuning results are described in Table 14 and Table 15 for ARCD dataset and TyDi QA dataset respectively. AraBERT achieves the highest F1 and EM scores on both evaluation datasets. Compared to the previous task, monolingual models do not show a significant advantage over multi-lingual BERT. The results are highly comparable, especially with ARBERT, MARBERT, and mBERT. The authors of ARBERT and MARBERT have also pointed this out, and this could be because these models have been pre-trained with a maximum sequence length of 128 tokens. On the Tydi QA dataset, AraBERT achieves the highest F1 and EM scores. Based on these results, AraBERT is the best performing model for reading comprehension.

Model	F1	Exact Match
mBERT	52.845	21.22
XLM-Roberta	24.88	7.83
ARBERT	54.43	22.50
MARBERT	50.89	20.37
AraBERT	**57.25**	23.50

Table 14: Fine-Tuning results on ARCD dataset for the task of MRC.

Model	F1	Exact Match
mBERT	55.57	41.69
XLM-Roberta	32.32	19.86
ARBERT	52.82	33.33
MARBERT	55.86	41.36
AraBERT	**67.611**	**49.511**

Table 15: Fine-Tuning results on TyDi QA dataset for the task of MRC.

4.5.4 Experiment 4: Adapter-Tuning on Reading Comprehension Task

Like PR, a task adapter is applied to each model and fine-tuned on the SQuAD 1.0 dataset while freezing the original parameters. The results of adapter-tuning on ARCD and TyDi QA datasets are described in Table 16 and Table 17 respectively. AraBERT again outperforms all models on both ARCD and Tydi QA datasets. Compared to fine-tuning most models actually increase in performance compared to fine-tuning. Figure 22 shows impact of adapter-tuning compared to fine-tuning on mBERT. The performance increases when using adapter-tuning on the Tydi QA dataset. Unlike mBERT, XLM-Roberta shows a decline in performance on both datasets as shown in Figure 23a for ARCD dataset and in Figure 23b for the Tydi QA dataset. ARBERT performs better with adapter-tuning on Tydi QA while slightly worse on ARCD. The results of ARBERT on ARCD and TyDi QA are in best described in Figure 24a and Figure 24b, respectively. MARBERT performs slightly worse with adapter-tuning, while AraBERT performs better using adapter-tuning when evaluated on Tydi QA. The results of MARBERT on ARCD and TyDi QA datasets are shown in Figure 25a and Figure 25b,

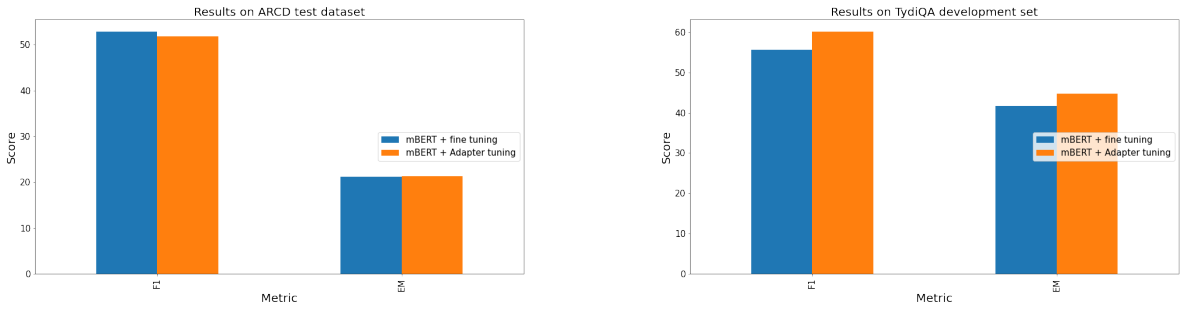
respectively. The results of AraBERT on ARCD are shown in Figure 26a while the results on Tydi QA are shown in Figure 26b. While most models improve on the Tydi QA dataset, most models drop in performance when evaluated on ARCD. As indicated in previous sections, ARCD is a more challenging dataset, and the drop in EM is high across all models apart from ARBERT, which performs better with adapters.

Model	F1	Exact Match
mBERT	51.81	21.36
XLM-Roberta	23.83	7.69
ARBERT	53.48	22.79
MARBERT	47.467	17.23
AraBERT	**56.27**	**22.93**

Table 16: Adapter-Tuning results on ARCD dataset for the task of MRC.

Model	F1	Exact Match
mBERT	60.17	44.733
XLM-Roberta	27.68	16.72
ARBERT	62.26	46.68
MARBERT	53.41	38.11
AraBERT	**66.58**	**49.51**

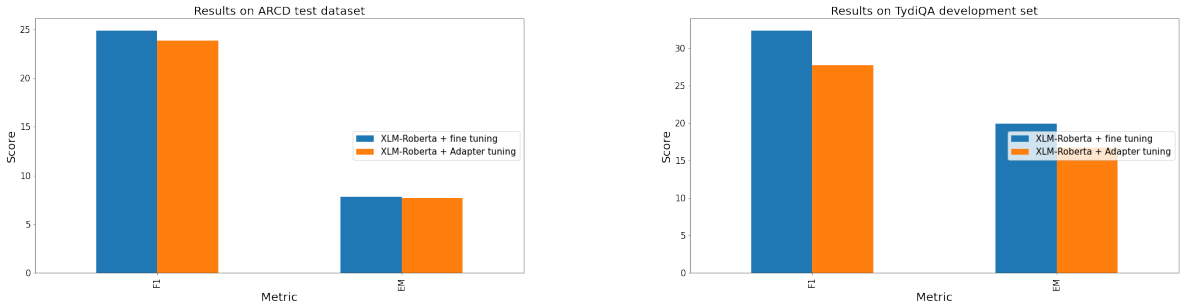
Table 17: Adapter-Tuning results on TyDi QA dataset for the task of MRC.



(a) ARCD test dataset

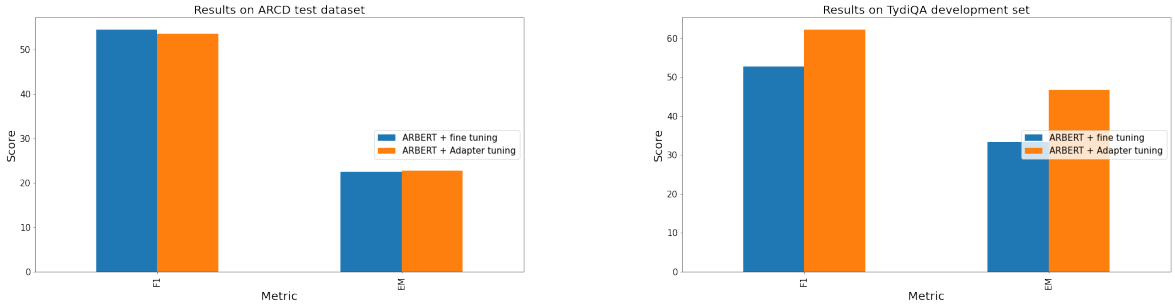
(b) Tydi QA dataset

Figure 22: results of fine-tuning mBERT and adapter tuning of mBERT (a) ARCD (b) Tydi QA



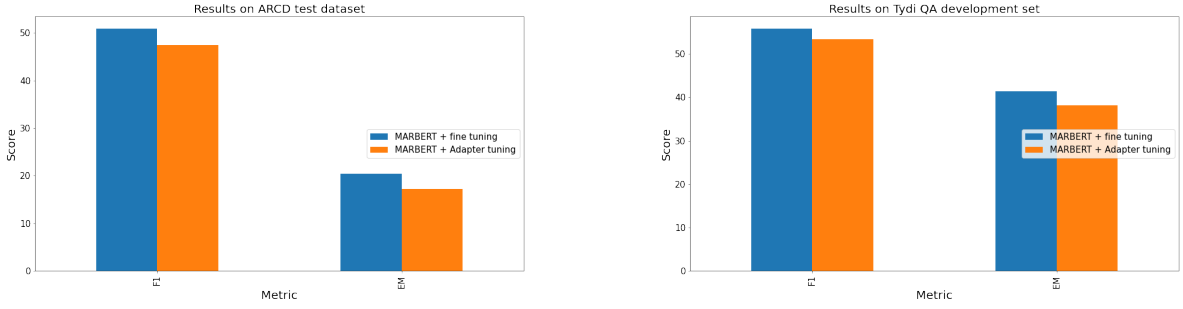
(a) ARCD dataset (b) Tydi QA test dataset

Figure 23: results of fine-tuning XLM-Roberta and adapter tuning of XLM-Roberta (a) ARCD (b) Tydi QA



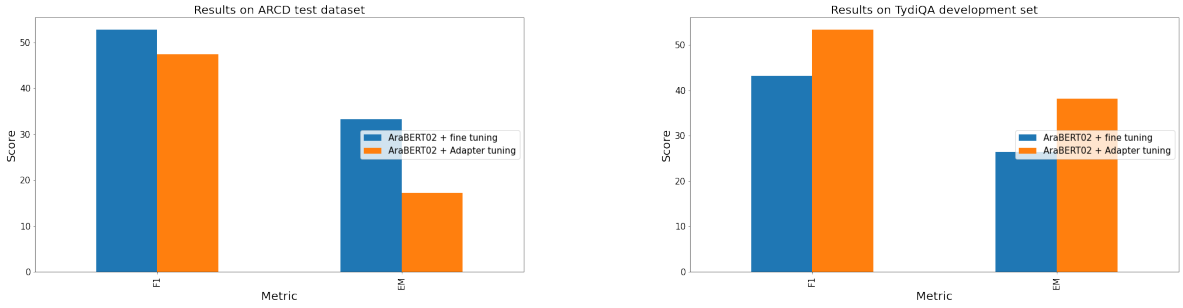
(a) ARCD dataset (b) Tydi QA test dataset

Figure 24: results of fine-tuning ARBERT and adapter tuning of ARBERT (a) ARCD (b) Tydi QA



(a) ARCD dataset (b) Tydi QA test dataset

Figure 25: results of fine-tuning MARBERT and adapter tuning of MARBERT (a) ARCD (b) Tydi QA



(a) ARCD dataset (b) Tydi QA test dataset

Figure 26: results of fine-tuning AraBERT and adapter tuning of AraBERT (a) ARCD (b) Tydi QA

5 Conclusion

The current work evaluates parameter efficient transfer learning and fine-tuning techniques for Arabic QA tasks, including passage re-ranking and reading comprehension tasks. Our study explores two families of language models, namely multi-lingual and monolingual. While cross-lingual transfer learning performs better than the BM25 baseline in passage re-ranking, monolingual models are far superior in performance. As indicated in [13], the ARCD dataset contains more challenging questions and better performance indicators than TyDi QA. Based on the results, monolingual Arabic models generalize better on different benchmarks than multi-lingual models. ARBERT performs the best on all metrics for the re-ranking task and can act as a baseline for future studies in Arabic IR. On the other hand, AraBERT achieves the best results on MRC, which is consistent with the results in [14] and therefore is recommended as a baseline for MRC. For cross-lingual models, mBERT results in MRC are very close to the monolingual models suggesting further research in Arabic MRC. Progress in Arabic MRC systems is still lagging behind their English counterparts. As for adapter tuning, the results confirm that parameter efficient transfer learning reduces trainable parameters without compromising performance significantly. Adapter tuning proves to be a viable light-way alternative to fine-tuning while keeping performance relatively high. This also suggests that adapters can enhance the performance of certain language models on low resources languages, as described in the results. Depending on the problem, adapters can effectively replace fine-tuning methods and allow for better portability and sharing of modules across different tasks. However, it is unclear why specific language models improve while others slightly decline. In the case of MRC, parameter efficient transfer learning improves the performance of several models on Tydi QA and shows a drop in EM on the ARCD dataset. Based on the problem and application, this drop could be significant and not acceptable. If extremely high accuracy is not the priority, adapter-tuning is highly recommended, especially in cases where multiple models are needed for fine-tuning.
Below are the answers to research questions addressed in this study:

Q1. *How does monolingual transfer learning perform in a re-ranking task in comparison with the baseline ranker BM25 and to cross-lingual transfer learning?*

The experiments show that monolingual models such as ARBERT and MARBERT can outperform the baseline BM25 in the Arabic passage re-ranking task. The performance of monolingual models is far more superior to cross-lingual models in passage re-ranking. Both ARBERT and MARBERT are suitable strong baselines for Arabic IR tasks.

Q2. *How effective is monolingual transfer learning in reading comprehension tasks for the Arabic language compared to cross-lingual transfer learning?*

The result of the MRC task suggests that multi-lingual models can achieve close results to the best performing monolingual models. The performance of mBERT is comparable to the monolingual models, suggesting that Arabic monolingual models should be further investigated. The existing Arabic monolingual models are still lagging behind English models in MRC.

Q3. *Can parameter efficient transfer learning match the performance of fine-tuning entire models for low resource languages such as Arabic in QA tasks?*

Yes, the results achieved show comparable results with fine-tuning entire models. For MRC, adapter tuning further improves the performance and achieves better results than full fine-tuning.

Q4. *How does parameter efficient transfer learning perform when applied on monolingual models compared to cross-lingual models ?*

The behavior of adapter-tuning on monolingual and multi-lingual models is very similar. Adapter-tuning improves mBERT, ARBERT and MARBERT on MRC on TydiQA dataset compared to

fine-tuning. The decline in performance is also apparent in both language models on passage re-ranking tasks. In general, both monolingual and multi-lingual models can benefit from adapters as an alternative to fine-tuning.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Statista, "Share of the most common languages on the internet – statista," 2021. Online; accessed 12-December-2021,.

[3]

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[5] O. Khattab and M. Zaharia, "Colbert: Efficient and effective passage search via contextualized late interaction over BERT," *CoRR*, vol. abs/2004.12832, 2020.

[6] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," 2021.

[7] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman, "Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?," 2020.

[8] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho, and I. Gurevych, "Adapterhub: A framework for adapting transformers," 2020.

[9] D. Jurafsky and J. H. Martin, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson Prentice Hall, 2021.

[10] C. Zeng, S. Li, Q. Li, J. Hu, and J. Hu, "A survey on machine reading comprehension: Tasks, evaluation metrics and benchmark datasets," 2020.

[11] M. Alkhatnai, H. Amjad, M. Amjad, and A. Gelbukh, "Methods and trends of machine reading comprehension in the arabic language," *Computación y Sistemas*, vol. 24, 12 2020.

[12] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," *CoRR*, vol. abs/1902.00751, 2019.

[13] H. Mozannar, K. E. Hajal, E. Maamary, and H. Hajj, "Neural arabic question answering," 2019.

[14] M. Abdul-Mageed, A. Elmadany, and E. M. B. Nagoudi, "Arbert marbert: Deep bidirectional transformers for arabic," 2021.

[15] Wikipedia contributors, "Sigmoid function — Wikipedia, the free encyclopedia," 2021. [Online; accessed 28-December-2021].

[16] Wikipedia contributors, "Rectifier (neural networks) — Wikipedia, the free encyclopedia," 2021. [Online; accessed 28-December-2021].

[17] B. Green, K. Alice, Chomsky, Carol, Laughery, and Kenneth, "Baseball: An automatic question answerer," vol. 19, pp. 219–224, 05 1961.

[18] W. Woods, *Lunar Rocks in Natural English: Explorations in Natural Language Question Answering*, vol. 5, pp. 521–569. 01 1977.

[19] A. Holst, "Total data volume worldwide 2010-2025," Jun 2021. Online; accessed 5-December-2021.

[20] Microsoft, "MS MARCO." Online; accessed 5-December-2021.

[21] W. contributors, "Arabic language — wikipedia, the free encyclopedia," 2004. Online; accessed 22-November-2021 last edited on 30 November 2021.

[22] W. contributors, "Spread of islam — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 2 December 2021.

[23] W. contributors, "Languages used on the internet — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 12 November 2021.

[24] A. of W3Techs, "World wide web technology surveys," 2021. Online; accessed 12-December-2021.

[25] W. contributors, "Arabic diacritics — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 5 December 2021.

[26] W. contributors, "Varieties of arabic — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 24 November 2021.

[27] K. Shaalan, S. Siddiqui, M. Alkhatib, and A. Monem, *Challenges in Arabic Natural Language Processing*, pp. 59–83. 11 2018.

[28] E. Voorhees and D. Harman, "Proceedings of the eighth text retrieval conference (trec-8)," 11 2000.

[29] B. R. Rowe, D. Wood, A. Link, and D. A. Simoni, "Economic impact assessment of nist's text retrieval conference (trec) program. final report," 2010.

[30] C. Peters and M. Braschler, "Cross-language system evaluation: The clef campaigns," *JASIST*, vol. 52, pp. 1067–1072, 10 2001.

[31] F. Gey and D. Oard, "The trec-2001 cross-language information retrieval track: Searching arabic using english, french or arabic queries.," 01 2001.

[32] R. Besançon, S. Chaudiron, D. Mostefa, O. Hamon, I. Timimi, and K. Choukri, "Overview of clef 2008 infile pilot track," vol. 1174, pp. 939–946, 09 2008.

[33] "Boolean model of information retrieval," Sept. 2021. Page Version ID: 1043894312.

[34] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11–21, 1972.

[35] M. M. Beaulieu, M. Gatford, X. Huang, S. Robertson, S. Walker, and P. Williams, "Okapi at trec-5," in *The Fifth Text REtrieval Conference (TREC-5)*, pp. 143–165, Gaithersburg, MD: NIST, January 1997.

[36] W. contributors, "Tf-idf — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 18 September 2021.

[37] W. contributors, "Okapi bm25 — wikipedia, the free encyclopedia," 2004. Online; accessed 12-December-2021, last editied 24 February 2021.

[38] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends in Information Retrieval*, vol. 3, pp. 333–389, 01 2009.

[39] S. Wang, M. Yu, J. Jiang, W. Zhang, X. Guo, S. Chang, Z. Wang, T. Klinger, G. Tesauro, and M. Campbell, "Evidence aggregation for answer re-ranking in open-domain question answering," 2018.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[41] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for arabic language understanding," 2021.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[43] A. Alishahi, G. Chrupała, and T. Linzen, "Analyzing and interpreting neural networks for nlp: A report on the first blackboxnlp workshop," 2019.

[44] M. de Lhoneux, J. Bjerva, I. Augenstein, and A. Søgaard, "Parameter sharing between dependency parsers for related languages," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 4992–4997, Association for Computational Linguistics, Oct.-Nov. 2018.

[45] P. Lewis, B. Oğuz, R. Rinott, S. Riedel, and H. Schwenk, "Mlqa: Evaluating cross-lingual extractive question answering," 2020.

[46] Y.-H. Lin, C.-Y. Chen, J. Lee, Z. Li, Y. Zhang, M. Xia, S. Rijhwani, J. He, Z. Zhang, X. Ma, A. Anastasopoulos, P. Littell, and G. Neubig, "Choosing transfer languages for cross-lingual learning," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 3125–3135, Association for Computational Linguistics, July 2019.

[47] L. Mason, P. Bartlett, and M. Frean, "Boosting algorithms as gradient descent in function space," 06 1999.

[48] A. Conneau, 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," 2020.

[49] Y. Zhou, X. Geng, T. Shen, W. Zhang, and D. Jiang, "Improving zero-shot cross-lingual transfer for multilingual question answering over knowledge graph," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 5822–5834, Association for Computational Linguistics, June 2021.

[50] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[51] L. Pratt and S. Thrun, "Inductive transfer. 2nd special issue," *Machine Learning*, vol. 28, 01 1997.

[52] J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, p. 149–198, Mar 2000.

[53] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," 2019.

[54] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[55] X. Liu, P. He, W. Chen, and J. Gao, "Improving multi-task deep neural networks via knowledge distillation for natural language understanding," 2019.

[56] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[57] G. Lample and A. Conneau, "Cross-lingual language model pretraining," *CoRR*, vol. abs/1901.07291, 2019.

[58] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[59] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[60] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, PMLR, 09–15 Jun 2019.

[61] M. E. Peters, S. Ruder, and N. A. Smith, "To tune or not to tune? adapting pretrained representations to diverse tasks," 2019.

[62] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "Mad-x: An adapter-based framework for multi-task cross-lingual transfer," 2020.

[63] Y. Liu, 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," 2019.

[64] L. H. Bonifacio, I. Campiotti, R. Lotufo, and R. Nogueira, "mmarco: A multilingual version of ms marco passage ranking dataset," 2021.

[65] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016.

[66] J. H. Clark, E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev, and J. Palomaki, "Tydi QA: A benchmark for information-seeking question answering in typologically diverse languages," *CoRR*, vol. abs/2003.05002, 2020.

[67] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, pp. 1532–1543, 01 2014.

[68] K. M. Hermann, T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," 2015.

[69] A. Virtanen, J. Kanerva, R. Ilo, J. Luoma, J. Luotolahti, T. Salakoski, F. Ginter, and S. Pyysalo, "Multilingual is not enough: Bert for finnish," 2019.

[70] S. Dadas, M. Perełkiewicz, and R. Poświata, "Pre-training polish transformer-based language models at scale," 2020.

[71] "Python elasticsearch client¶."

[72] E. Voorhees, "The trec-8 question answering track report," 11 2000.