Arab American University - Ramallah

Faculty of Graduate Studies

Blood Disease Prediction from Arabic Medical Dataset Using Arabert through Name Entity Recognition

By

Murad Halabouni

Supervisor

Prof. Osama Mansour

This thesis was submitted in partial fulfillment of the requirements for the Master`s degree in

Cybercrime and Digital Evidence Analysis.

November / 2022

Blood Disease Prediction from Arabic Medical Dataset Using Arabert through Name Entity Recognition

By

Murad Halabouni

This thesis was defended successfully on January 5th, 2023 and approved by:

Committee members		Signature
1. Supervisor	Prof. Osama Mansour	
2. Internal Examiner Name	Dr. Huthaifa Ashkar	
3. External Examiner Name	Dr. Catherine Abu Kwaik	

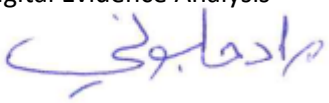
Declaration

I Murad Halabouni, declare that work presented in this dissertation is my own. The thesis titled with "Blood Disease Prediction from Arabic Medical Dataset Using Arabert through Name Entity Recognition" contains no material that has been submitted previously, in whole or in part, for the award of any other academic degree or diploma. Except where otherwise indicated, this thesis is my own work.

Murad Halboni

201920301

Cybercrimes and Digital Evidence Analysis

201920301 

Date: 14 / 5 / 2023

Abstract

Natural Language Processing (NLP) tasks have been showing state-of-the-art results since the appearance of Bidirectional Encoder Representations from Transformers (BERT). However, literature review shows that BERT performs very effectively at modeling domain-specific datasets when Fine-Tuned and modified. This was not the case before BERT since specific-domain datasets were very challenging to NLP task due to the type of these datasets where they are usually rich with scientific and technical jargon. This thesis will use AraBERT, a special variant of BERT that is trained to understand Arabic language. AraBERT will be used in a different context by altering and modifying the model to help us achieve a successful prediction from a specific-domain Arabic dataset and answer our research questions. The result of this work shows big improvements in prediction accuracy for specific-domain Arabic medical dataset. The final results demonstrate the model's capability and robustness to predict blood disease symptoms.

Acknowledgements

I would like to thank my esteemed supervisor Prof. Osama Mansour for his valuable supervision and support during the course of my thesis.

My gratitude extends to my University ARAB AMERICAN UNIVERSITY AAUP for providing such academic programs that help us to change the future for a better one.

Finally, my appreciation goes out to my family and friends for their encouragement and support.

Table of Contents

List of Tables

List of Equations

List of Figures

CHAPTER ONE
INTRODUCTION

1.1 Background

Electronic information is becoming increasingly significant and abundant in various fields, such as medical data for example. The ease of access and participation led to massive materials in different forms such as medical and clinical reports, articles, online platforms that have qualified doctors writing articles and providing guidance, and now official institutions are depending on electronic systems to provide structured data related to admitting patience, diagnosing process, writing prescriptions, etc. The availability of such data allows for clinical knowledge extraction, where causes and answers can be concluded. (Dai et al., 2019)

In order to extract information, modern machine learning algorithms are being used and implemented, one of the most effective and proven successful Branch is Natural Language Processing (NLP). NLP facilitates knowledge extraction and provides insight for professionals in an automated way. NLP has many subtasks, one of them is Named Entity Recognition (NER). NER allows to mark and identify words or chunks of words that refer to a specific entity. For example, entities can be symptoms, medical terminologies, diseases names, or any entity that may be needed. These identified entities can be extracted from massive data, which then can be reported to experts which gives them knowledge to take action. (Yadav & Bethard, 2019).

To achieve state-of the-art results in our NER task, we will use and dive into Bidirectional Encoder Representations from Transformers known as “BERT”, BERT was published in 2017 by researchers at Google AI Language and caused a stir in Machine learning community. BERT is based on Transformers, which represents a deep learning model that finds dependencies between input and output elements. BERT

model was pre-trained with very large corpus using data extracted from English Wikipedia with 2500M word and BooksCorpus with 800M words, and that has made BERT very efficient with Language Understanding and gave state-of-the-art results for most of NLP Tasks. BERT was pre-trained on non-specific domain data, and so, a number of variants surfaced based on BERT where BERT variants are pre-trained on a specific domain dataset. (Vaswani et al., 2017).

In 2021, three researchers from American University of Beirut pre-trained BERT to understand Arabic language and called the variant AraBERT. The results of AraBERT were state-of-the-art which allowed for numerous applications to use Arabic language in machine learning modeling. Literature review showed that Arabic language suffers from the lack of specific domain datasets where most of the existing Arabic datasets are general. With BERT, it was understandable that it is difficult and consuming to design algorithms to understand Arabic, not to mention perform various tasks of any kind. (Antoun et al., 2020).

In this thesis, a new Arabic Medical Dataset is built which has been gathered from reliable medical resources. After creating this dataset, it is used to pre-train AraBERT to perform Natural Language Processing “NLP” subtask, Named Entity Recognition “NER”.

1.2 Problem Statement

Language models which are built for specific domain purposes are required to have more accurate and reliable predictions and results. This task can be done with the help of specific-domain dataset that are built for specific tasks, in our case a disease prediction in Arabic language. Due to the complex nature of medical text where its syntax contains specific terminologies that are uncommon and hard to read, this made

it difficult for general language models to perform a prediction with accurate results from Arabic datasets or even worse they give a false result. Researchers who used machine learning models that understand Arabic Language to perform NLP tasks using specific-domain datasets faced a problem locating accurate Data, for instance the recent research done by (Abdul et al., 2021) could not release their Dataset in order to avoid any copy-right issues. Despite sharing their model online.

In modern language models, we can find that Arabic is almost ignored when compared to English. (Antoun et al., 2020) in their work tried to create Arabert, and they expressed the difficulty they had to collect Arabic resources "relatively few resources and a less explored syntax compared to English" even though Arabic is morphologically very rich which make it very challenging to work with. With BERT, Arabic is now understandable by language models, which opens the path for all different tasks and a new variant of BERT surfaced that deals with Arabic language to perform general domain tasks. (Abdelali et al., 2020.)

Hence, we will tackle lack of resources for specific-domain Arabic dataset by creating Dataset from scratch, by collecting Blood disease symptoms and reports, then perform pre-processing step on data and locate features. Then we will use AraBERT in a different context by altering and modifying the model to help us achieve a successful prediction from a specific-domain Arabic dataset. The model will allow us to perform prediction, also the model will be fine-tuned, for the purpose of achieving a successful prediction. So due to lack of specific-domain Arabic dataset, and the absence of trained language model that understand Arabic text in a specific-domain, we formulated our research questions:

Question 1: Does AraBERT fine-tuning achieve a successful prediction using specific-domain Arabic dataset?

Question 2: Does AraBERT fine-tunning achieve higher accuracy compared to another model that uses Arabic dataset?

Question 3: Does creation of specific-domain dataset help the model to achieve successful and more accurate prediction?

Question 4: Can our model perform a prediction for another domain, e.g., criminal domain?

1.3 Thesis Outline

This thesis is structured as follows:

- **Chapter 1, Introduction.** An overview of the topic we are discussing in this thesis and an explanation of our problem statement with the thesis question.
- **Chapter 2, Literature Review.** This chapter introduces the previous approaches and work that led to BERT and elaborates on BERT components. Then, Nature Language Process is introduced along with all technical concepts, and Name Entity Recognition is added to complete all technical concepts.
- **Chapter 3, Research Methodology.** Domain-Specific BERT is introduced, followed by our model AraBERT. Then, the methodology flowchart of our work is introduced, an elaboration of how we collected and created our dataset. Finally, the implementation of a fine-tuning process and how to evaluate the model.
- **Chapter 4, Result.** Results and discussion have been mentioned in this chapter,

- **Chapter 5, DISCUSSION AND BENCHMARKING.** This reflects the key observations and insights from using AraBERT to perform NER in Arabic, then a comparison with recent benchmarking studies.
- **Chapter 6, Conclusion.** We answer our research questions, and also discuss challenges and limitations observed.

CHAPTER TWO
LITERATURE REVIEW

The theoretical foundations of this thesis are introduced in this chapter. Starting with the approaches and developments that led to BERT. Followed by the technical concept of Natural Language Processing, Named Entity Recognition, and Feature Representations.

2.1 Towards Bert

There are a number of technologies that have a great effect on the emergence of BERT. BERT also has an effect on natural language processing research in returns, as the following subsection will show.

2.1.1 Long-Short Term Memory and ELMo

Benchmarks research that used deep learning models for the purpose of NLP was mainly focused on using Recurrent Neural Networks (RNN). Due to the importance of word order in establishing sentence meaning in NLP tasks, recurrent neural network seemed to be well-suited. RNN has correlation regarding time-steps, where the current time-step state is dependent on the previous time-steps. On the contrary, other standard feed-forward networks, where the flow of information is forward state only, they are clueless about previous time-steps. The sequential nature of RNN has a downside as it cannot work in parallel, which is translated to making RNN training much more computationally expensive. Also make them more vulnerable to forget what previously has been learnt, since long-term dependencies tend to be forgotten due to decaying in gradients (Hochreiter & Schmidhuber, 1997).

In order to overcome vanishing and exploding gradients issue which is the result of having a relatively small output from certain layer compared to the value of the input at the beginning of layer. this behavior is less frequent when using shallow network where there are a few layers used, while it has more affect when number of layers used is more relatively.

LSTM was developed. Long-Short Term Memory (LSTM) is based on RNN, where it replaces time-step with memory cell and series of gates, these gates help memory cells to decide if to keep information or discard them and this made LSTM capable of maintaining long term dependencies. Many variants were built from LSTM, and the most effective and must mention variant is bidirectional LSTM (BiLSTM). BiLSTM achieved state-of-the-art results, where it concatenates two LSTM networks. The first LSTM takes input from front to back, the second LSTM takes input from back to front, and the result is an encoded vector containing both outputs.

Embedding from Language Models "ELMO" which was created by BiLSTM, is a language model trained on a huge corpus. ELMo is the first Bidirectional language model that can mitigate many NLP problems such as polysemy where words can have different context meaning and misunderstanding. ELMo allows to put a word in context using vector representation where the same word have different vector representations. Vector representation of ELMo is character based while its precedents are word-based, and this allows it to mitigate out-of-vocabulary (OOV) issue by representing ambiguous words as a representation of their most frequent occurring characters. ELMo demonstrated the advantages of pre-training and fine-tuning language model to perform downstream tasks.(Hochreiter & Schmidhuber, 1997; Cho et al., 2014)

2.1.2 Encoder-Decoder, Attention, Self-Attention and Transformer (attention is all you need)

In sequence-to-sequence modeling such as in machine translation tasks, encoder-decoder RNN was the ideal structure till the development of transformers. An input sequence is fed to the encoder to be transformed from one language to a vector representation, which will be the input to the decoder side to be converted into an output sequence in another language. Figure 2.2 exhibit components of Encoder-Decoder Recurrent Neural Network.

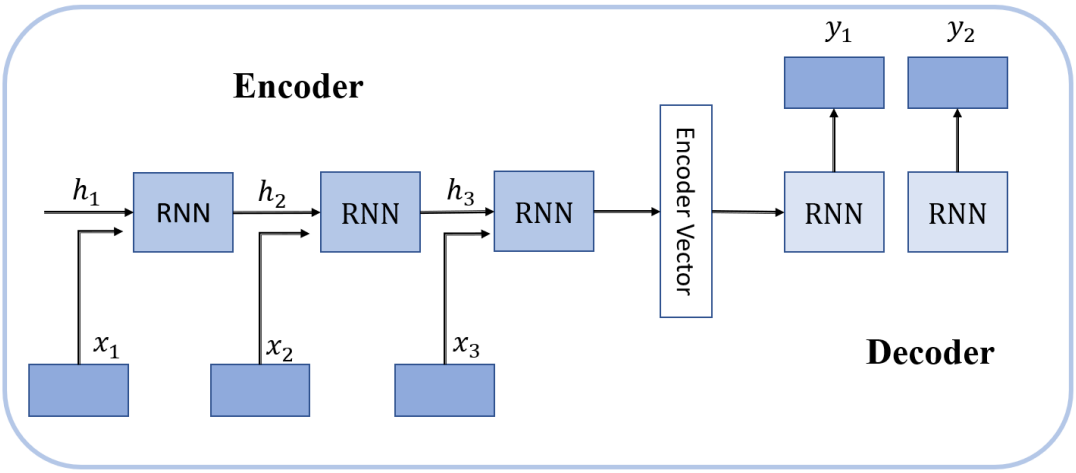


Figure 2. 1 Encoder-Decoder Recurrent Neural Network. (Jay Alammar., 2019)

With the advancement of RNN, there is still a major problem especially with long words and here come the transformers with its ability to process text as whole (Vaswani et al., 2017). That means that the transformers will operate in a sequential order and process the input as one piece then produce a vector representation, and this allowed for the parallelization which resulted in improving the performance of a model (Sutskever et al., 2014).

To further explain and dive into transformer architecture, transformers are divided into encoder and decoder as shown in figure 2.2. Both encoder and decoder can operate as stand-alone but there will be deterioration in performance. Transformer architecture

depends on stacking, meaning that encoder and decoder consist of stacked layers which allows for sequence operations where each layer can perform different NLP task. To exemplify, inside an NLP pipeline the first layer can be assigned to find part-of-speech tags, the second layer to find dependencies, this can be extended according to pipeline need.

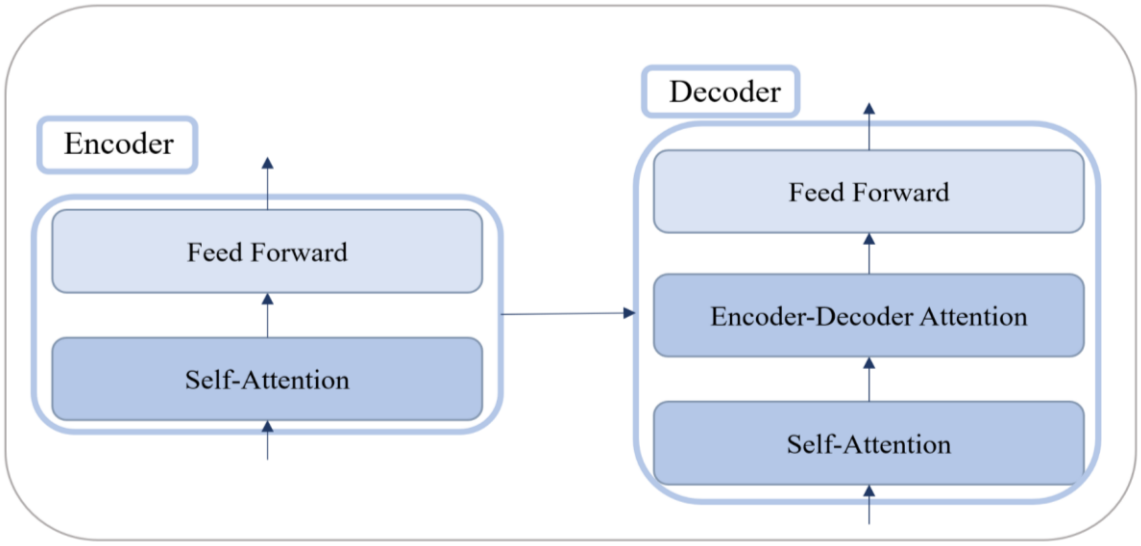


Figure 2. 2: Transformer Layers (Jay Alammar., 2019)

Encoders are built of two layers: Self-Attention and Feed Forward Neural Network. Before self-attention layer is the input data, a text for example that is converted to a numerical representation since the transformers will deal with the word's numerical representation not the word themselves. Now each word has its own representation or vector, this vector is inputted to the self-attention layer. The main functionality of the self-attention layer is to provide dependencies between different word vectors inside the same sequence. Feed-Forward Neural Network layer after self-attention layer plays the role of passing vectors onwards to the next encoder layer.

At this stage and due to the parallelization, a problem appeared where word order was lost, each word flow has its own calculated path which resulted in speeding up training considerably quickly on huge number of corpora. The solution for this issue

was to assign a unique ID representing the order of the word representation in a sentence. The result is a positional encoder vector. So down through different encoder layers word order is preserved (Devlin et al., 2018.).

Decoders on the other hand, have the same architecture of encoder, the only difference is the additional layer called encoder-decoder attention that located in between of self-attention layer and feed forward neural network layer. Decoders functionality is similar to encoders where vectors are processed in the same manner as encoders, and the additional attention layer is responsible to provide dependencies between different vector segments and layers. In addition, transformers architecture has the ability to perform multi-head attention, where multiple attention instances are computed between vectors at the same time, up to 12 attention head can be performed to create dependencies at the same time. This is translated to an excellent performance in training and results.

2.2 BERT

In this section, an introduction to BERT includes its definition, mechanism, architecture, and detailed implementation.

2.2.1 BERT Definition

BERT is an abbreviation for Bidirectional Encoder Representation from Transformer and is defined as a deep contextual language representation model developed by Google AI researchers. The main idea of designing BERT was to introduce a model capable of pre-training deep bidirectional representations of words from unlabeled text through joint conditioning on the left and right contexts simultaneously across all layers. And so, to fine-tune the pre-trained model, an output layer is added to provide high-

performing models for a variety of NLP applications, including text classification. Figure 2.3 next shows the learning steps of BERT, where in the first step the training is on large amount of dataset and in the second step the training is on a specific task with labeled dataset. (Devlin et al., 2018.).

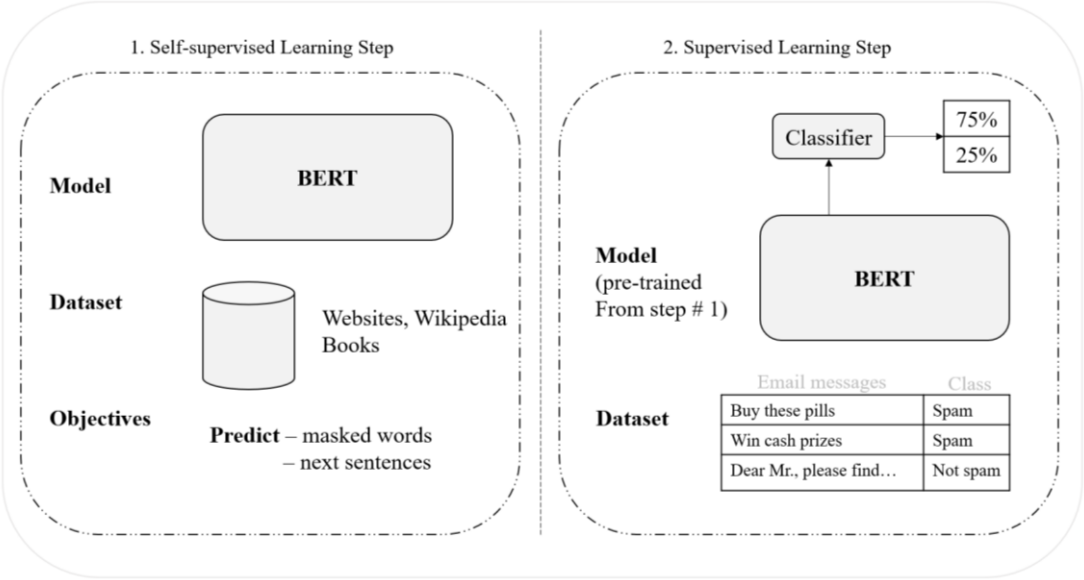


Figure 2. 3: BERT learning steps (Devlin et al., 2018)

One of the important concepts in BERT is self-attention. Self-attention considers a special form of attention that was initially introduced with the transformer model. Self-attention can be described as attention mechanism that links distinct points in a single sequence to calculate a contextual representation for each term in that sequence. Single self-attention is illustrated in figure 2.4 next which shows the attention on the word “it” from all the words in the sentence.

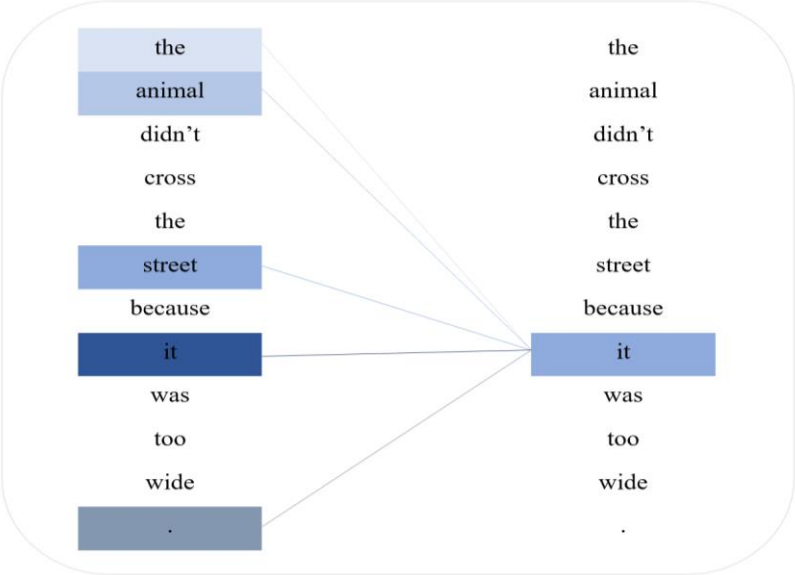


Figure 2. 4: Single Self-Attention Mechanism (Vaswani et al., 2017)

There is another mechanism which is multi-head attention. In multi-head self-attention instead of a single link, the link will be linearly projected number of times with different learned linear projections, and this enables jointly attention to information from several representation subspaces at different positions. (Devlin et al., 2018; Vaswani et al., 2017).

2.2.2 BERT Model Architecture

As indicated in the definition, BERT is a Bidirectional Transformer Encoder, which means it is composed of a stack of L identical transformer encoder layers, and in each layer, there are two sublayers. The first layer is a multi-head self-attention mechanism that allows encoding one word while looking at the other words. Position-wise fully connected Feed-Forward network (FFN) is the second layer, that is applied individually and identically to each location and contains two linear transformations, (Hendrycks and Gimpel, 2016). $(W_1 \in \mathbb{R}^{d_{model} \times d_{ff}}, b_1 \in \mathbb{R}^{d_{ff}}), (W_2 \in \mathbb{R}^{d_{ff} \times d_{model}}, b_2 \in \mathbb{R}^{d_{model}})$ such that

$$FNN(x) = max(0, xW_1 + b_1)W_2 + b_2 \quad (2.1)$$

Where d_{model} the dimensionality of input and output is, d_{ff} is the inner-layer dimensionality that is equal to $4d_{model}$. In BERT, the FFN using GELU activation function instead of ReLU, (Hendrycks and Gimpel, 2016) and it is defined as:

$$GELU(x) = 0.5x(1 + tanh(\sqrt{2/\pi}\ (x + 0.044715x^3))) \quad (2.2)$$

The encoder layer in BERT employs around each two sublayers a residual connection, then it is followed by normalization layer, where the output of each sublayer is defined as in equation and Sublayer(x) is representing the function implemented by the sublayer itself. (Hendrycks and Gimpel, 2016)

$$LayerNorm(x + Sublayer(x))$$

The sublayers will produce outputs of the same size of d_{model} in order to facilitate the residual connections. It is important to know that while the linear transformations are the same across positions within the same sublayer, BERT employs various parameters from layer to layer. There are two versions for BERT based on number of layers L, and this will specify the size of d_{model} and number of parameters:

1. BERT-base: $L = 12, d_{model} = 768, h = 12, d_{ff} = 3072$ (110M total parameters).
2. BERT-large: $L = 24, d_{model} = 1024, h = 16, d_{ff} = 408$ (340M total parameters).

The structure of BERT-base and the structure of a single encoder is illustrated in figure 2.5 next. (Vaswani et al., 2017) .(Devlin et al., 2018)

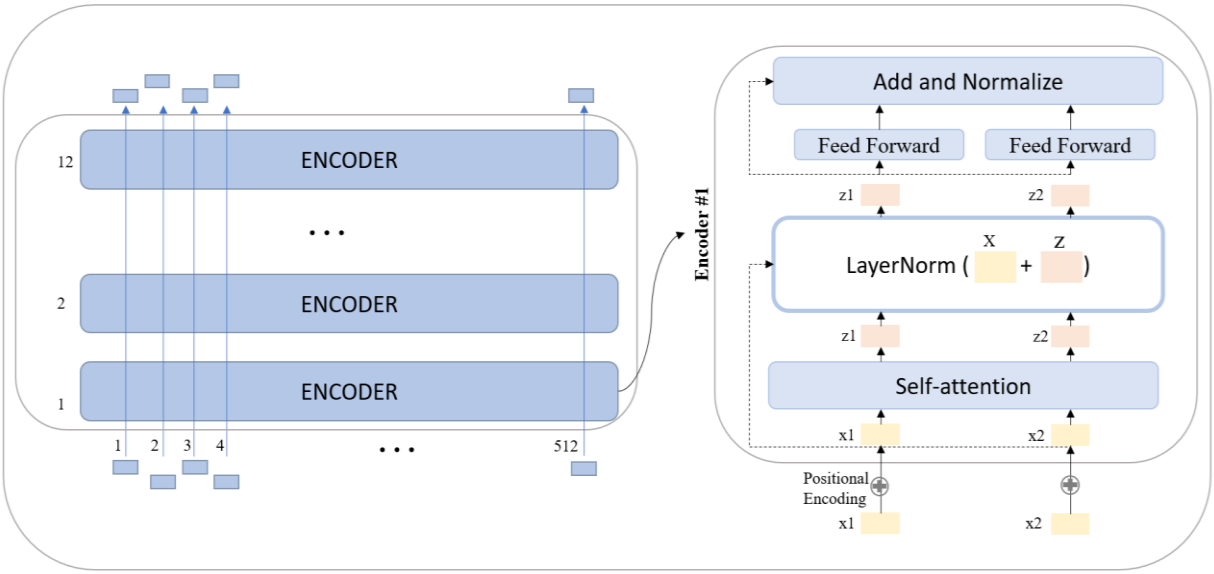


Figure 2. 5: Transformer Encoder Structure (Devlin et al., 2018)

2.2.3 Input Representation

In BERT the input is a sequence of words represented by what is called token (maximum is 512 tokens). The first step of BERT is to conduct a transformation on the words to provide numerical input representations for the model. To get input representations, three types of embeddings are combined together: token, segment, and positional embeddings as depicted in figure 2.6 next.

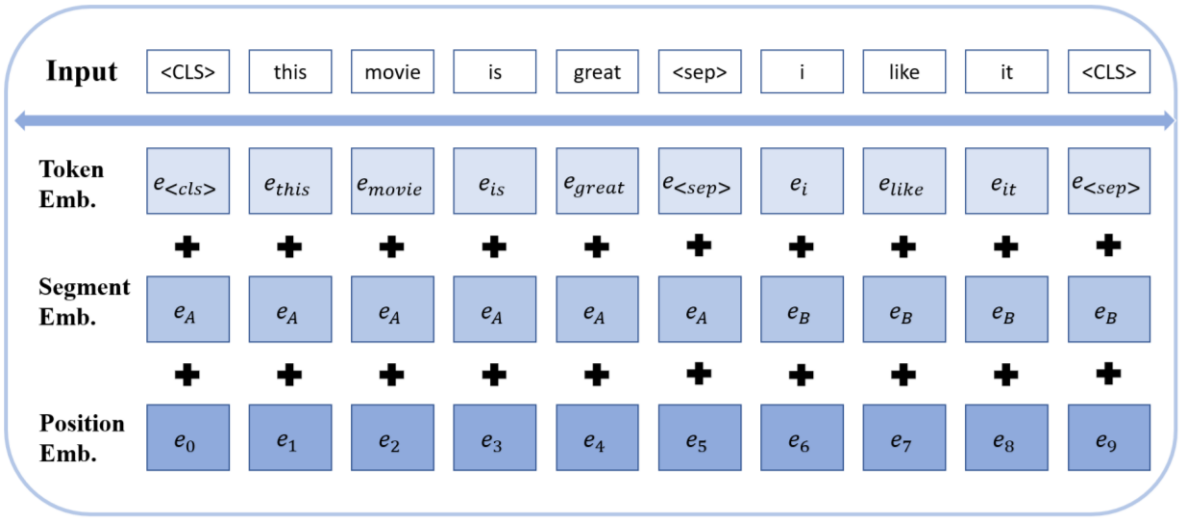


Figure 2. 6: The embeddings of the BERT input sequence (Devlin et al., 2018)

Three input types of BERT:

1. Token embedding: WordPiece embedding is used in BERT to tokenize the word in the input sequence. WordPiece is considered a model that creates fixed-size vocabularies of individual words that best fits a given language corpus. The tokenizer will check at first if the word is in its vocabulary, if the word does not exist, then it will break it into the largest possible subwords contained in the vocabulary. After the word being processed into one or more WordPiece tokens, the tokens will be used in order to restore the embeddings in the learned token embedding matrix.

BERT's vocabulary is the 30000 most common words found in English, all English characters, and three special tokens, which are:

A. [CLS]: CLS is a special token used at the beginning of every sequence.
B. [SEP]: SEP is used to differentiate between two sentences in case of multiple sentences passed as input, also, used at the end of the sequence.

As shown in the figure, we have two sentences: "This movie is great" and "I like it". These sentences will be tokenized by using tokenizer, then converted into numerical tokens, as follows:

['[CLS]', 'this', 'movie', 'is', 'great', '[SEP]', 'i', 'like', 'it', '[SEP]']

2. Segment embedding: This embedding is used in case we have more than one sentence. The learned segment will be added to each token indicating its sentence. This embedding has vocabulary of size 2. As shown in figure, the first words took symbol A, indicating sentence A and the last words took symbol B indicating the second sentence.

3. Positional embedding: This embedding is generated internally in BERT in order to provide information about the relative positions of the tokens in the input sequence.

Token, segment, and positional embeddings have same dimension of d_{model} ,so that, it will be easy to combine all these three embeddings together. (Devlin et al., 2018; Vaswani et al., 2017).

2.2.4 Pre-training Procedure

BERT is pre-trained simultaneously on two tasks. The pre-training was on a large corpus of unlabeled text from a book corpus (800 million words) and the entire Wikipedia (2500 million words). The pre-training methods are Next Sentence Prediction (NSP) and Masked Language Modeling (MLM).

- *Masked Language Modelling*

Masked Language Modeling MLM is the task of predicting a percentage of randomly masked input tokens, which is unlike Language Model that seeks to predict the next word given the sequence of preceding words.

Due to BERT's directionality, a masked language model was used to pre-train the model. The principal of this model is to feed the model with a sentence where 15% of the words in it are masked. The correct prediction of the masked words based on the context of the unmasked words in a sentence is what BERT does. Figure 2.7 next shows how MLM works where the output from the final encoder block will be fed into a classification layer representing FFNN and SoftMax in order to generate vocabularies and prediction of the masked words. Because the masked token will not be seen until

fine-tuning, the selected words are not always substituted with the [MASK] token and the loss function can consider only the predicted values for the masked words that make the learning more content based. (Devlin et al., 2018, Vaswani et al., 2017).

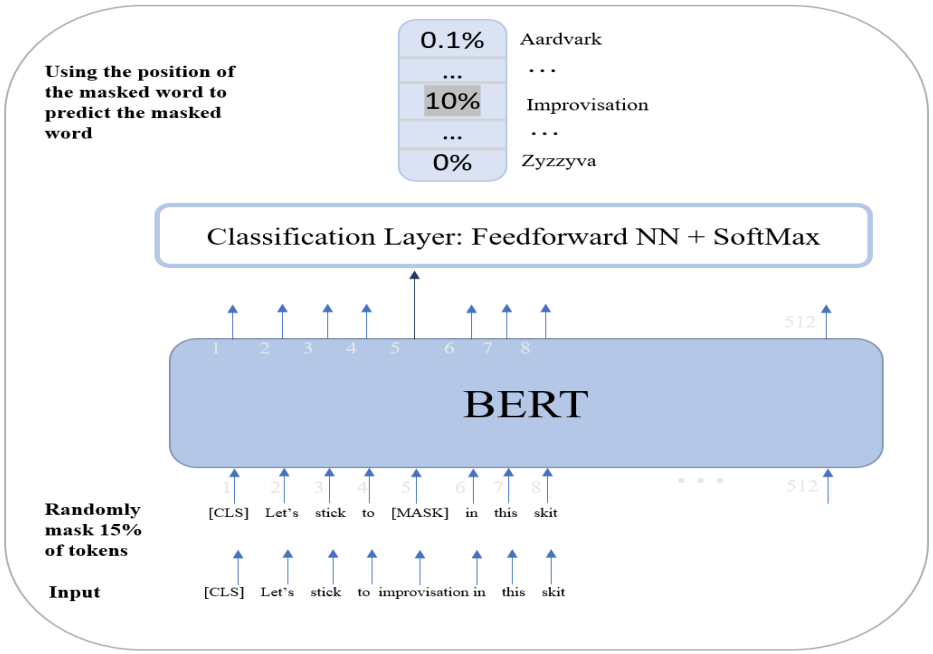


Figure 2. 7: Masked Language Model (Jay Alammar., 2019)

- *Next Sentence Prediction*

In this model, we have two sentences. NSP is a binary classification where its input is composed of two sentences and the prediction will occur in case the second sentence in the pair is the next sentence in the original graph. The main idea of this model is to help understanding the relationship between number of sentences that is not known by the language model, but it is highly significant for many downstream activities such as Natural Language Interface (NLI) and Question-Answering (QA).

The prediction in this model happens when selecting sentences, A and B for each pre-training example where B is chosen 50% of the time as the real next phrase that follows A (labelled as IsNext) and 50% of the time as a random text from the corpus (labeled as NotNext). In this scenario, as illustrated in figure 2.8 where the final hidden

vector corresponding to the [CLS] token is fed into an output SoftMax over the two possible predictions. (Mandelbaum & Shalev, 2016; Vaswani et al., 2017)

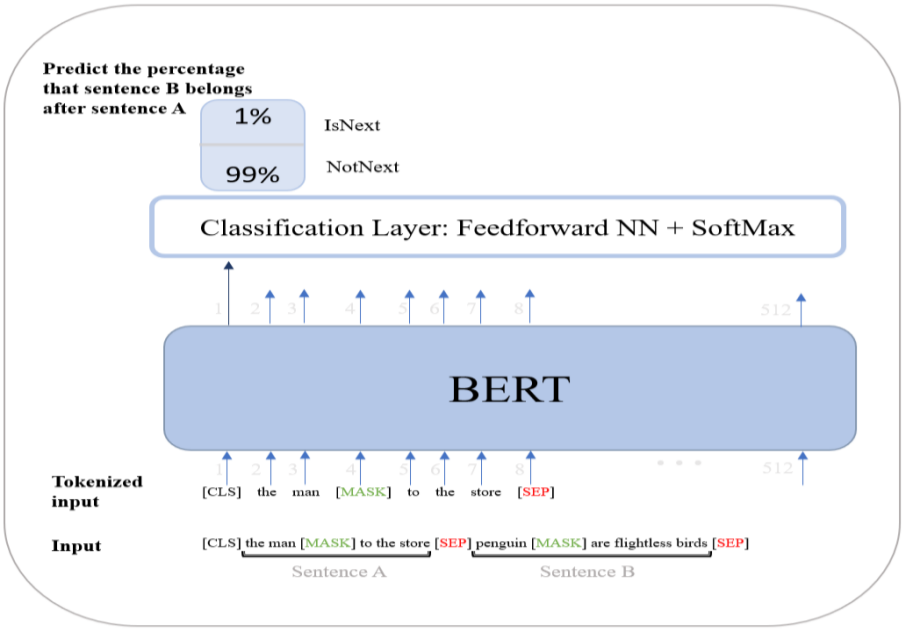


Figure 2. 8: Next Sentence Prediction (Jay Alammar., 2019)

2.2.5 Downstream Tasks

To apply the pre-trained language models to downstream natural language processing tasks, there are two approaches: Feature-based approach and fine-tuning approach. Feature-based approach employs task-specific structures with pre-trained representations as input features for task learning, but fine-tuning approach employs few task-specific parameters where the training is occurred on all pre-trained parameters through fine-tuning them. (Wu et al., 2016)

- *Fine-tuning Approach*

In BERT, its two pre-training objectives enable it to be deployed on any single sequence or sequence pair task without requiring significant task-specific architectural adjustments. In each task, task-specific inputs and outputs enter into BERT and fine-

tunes all of the parameters end-to-end for a few epochs. So, fine-tuning will add output layer to pre-trained model where all the parameters are jointly fine-tuned on downstream task as depicted in figure 2.9 next. BERT fine-tuning on many typical tasks. (Vaswani et al., 2017).

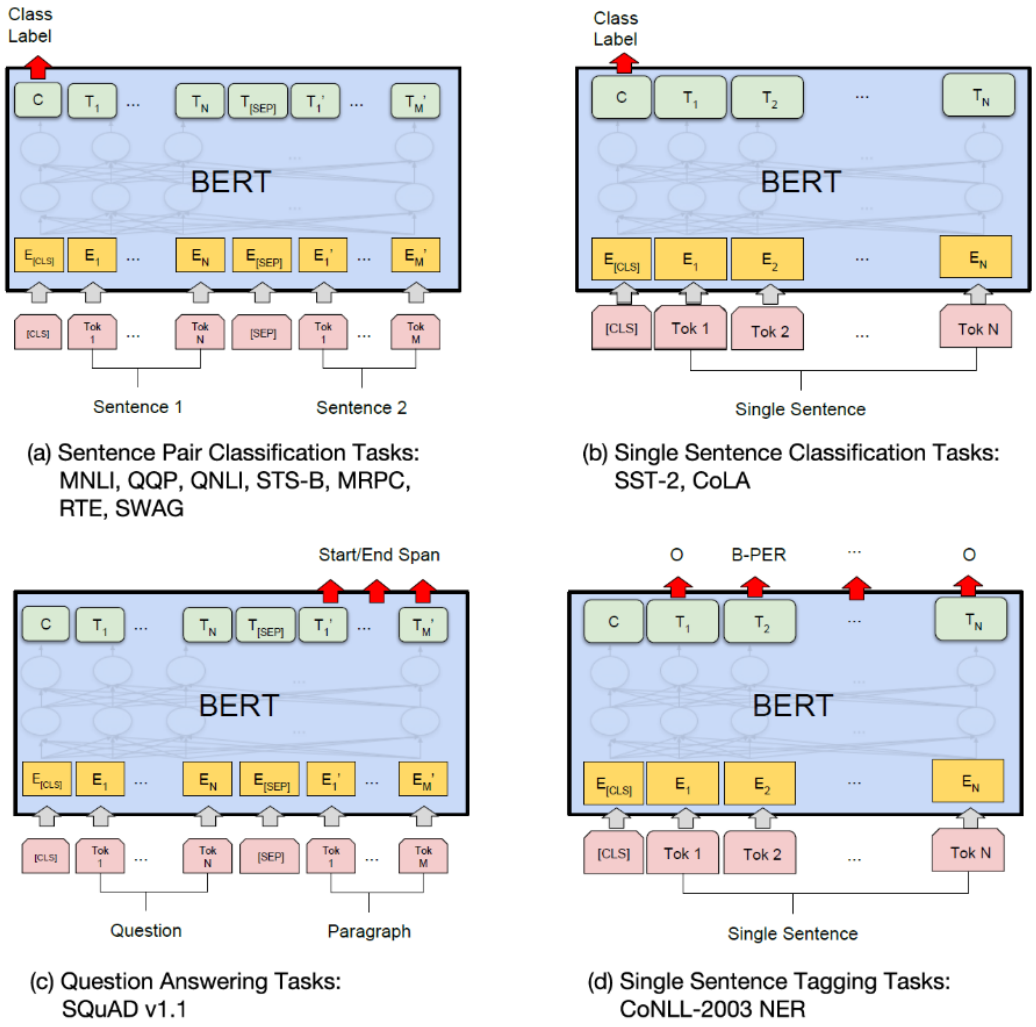


Figure 2. 9: BERT's fine-tuning on different task (Devlin et al., 2018)

- Task "a" and "b" are sentence classification, the difference is "a" has two sentences and "b" has one sentence. Model can distinguish this by the existence of [SEP] token. In case of a, where we need to perform a downstream task between the two sentences, Bert uses the embeddings learned from fine-tuning E1…En, and tries to produce a result or a

prediction based on the classifier. While in “b” the classifier needs one sentence to perform the task. An example of corpus and datasets used for fine-tuning different variant of BERT are Multi-Genre Natural Language Inference (MNLI), Quora Question Pairs (QQP), and Corpus of Linguistic Acceptability (CoLA).

- Task “c” is question answering exhibited is feeding Bert with a Question and a paragraph contains the answer. The classifier marks the beginning and end of the question tokens, and by comparing question tokens to tokens from the answer paragraph, the classifier marks the answer. The Stanford Question Answering Dataset s used to perform fine-tuning.
- Task “d” is question answering, the sentence tokens generated is used by the classifier to determine which entity the token belong to, based on the fine-tuning done. CoNLL-2003 is NER dataset contain English and German language.

- *Feature-based Approach*

The principle of Feature-based approach is to extract word representations from pre-trained model and use them as input for other task-specific structures. The importance of this approach comes from the tasks that cannot be simply represented by a transformer encoder structure, but need a task-specific structure, also to pre-compute a costly representation of the training data and then conduct multiple experiments with cheaper models on top of this representation has significant computational benefits (Vaswani et al., 2017)

2.2.6 Model Evaluation

The evaluation of the model is based on matching each token individually based on macro strategy that averages the individual scores. The concept of macro strategy is to deal with all entity types same by calculating the F-score for each class individually then averaging them. (Vaswani et al., 2017; van Barneveld & Le-Khac, 2016)

- *Confusion matrix*

Confusion matrix is considered a good way of showing accuracy, precision, recall, and f1-score metrics based on prediction labels in the model. The components of the confusion matrix are used to calculate the evaluation metrics, as shown next. (Devlin et al., 2018)

The components of the confusion matrix are:

- True Positive (TP) – the number of documents correctly classified as positive.
- False Positive (FP) – the number of positive documents predicted as negative.
- True Negative (TN) – the number of documents correctly classified as negative.
- False Negative (FN) – the number of negative documents predicted as positive.

Table 2. 1 Confusion Matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

For the evaluation indicators:

- **Accuracy**: The ratio of true predictions of the records, higher accuracy means better learning model prediction

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \quad (2.3)$$

- **Precision**: This metric indicates how many documents were truly correct from the testing documents.

$$Precision = \frac{TP}{TP+FP} \quad (2.4)$$

- **Recall**: Recall metric indicates how many documents from labeled dataset were identified as a match from the testing dataset.

$$Recall = \frac{TP}{TP+FN} \quad (2.5)$$

- **F1-Score (F1)**: This metric calculates the average of precision and recall in order to obtain a balanced value, higher F1 means a better learning model.

$$F1 = 2 * (\frac{Precision*Recall}{Precision+Recall}) \quad (2.6)$$

Later in 3.2.3 where Fine-Tunning is used, the prementioned evaluation indicators are the best suited to be used in this type of Predictions “Best Fit” (Bojanowski et al., 2016)

2.3 Natural Language Processing

This section elaborates on the Natural Language Processing (NLP) tasks and introduces mechanisms used by NLP to perform downstream tasks successfully. NLP uses different language models to create predictions or analyze documents and extract meanings from texts.

2.3.1 Natural Language Processing

Natural Language Processing is a disciplinary field and branch of computer science CS, artificial intelligence AI and Computational linguistics CL. it is concerned with using different computational methods in order to allocate and process statistical information related to a given text (Collobert et al., 2011; van Barneveld & Le-Khac, 2016). NLP uses number of computational methods to analyze texts and words segments to provide linguistic analysis. These techniques are designed to achieve human-like language processing for a variety of tasks or applications. The information in NLP is divided into two categories:

1. Semantic: a useful meaning is derived and extracted for a group of words or sentences, various methods are used to: locate name entities, decide the sentimental state of a word or a sentence and relation extraction.
2. Syntactic: the structure and basic building blocks of word or text is defined by using different methods called Part-of-speech-Tagging (POS-Tagging), these method segment words, tokenizing words and labeling each token.

Neural networks were used starting from the 2000s for the purpose of language modeling aiming to predict the next word in a text based on the previous one. In 2003, the first language model that used one hidden layer was introduced which is known later as word embedding (Bengio et al, 2014.). A word2Vec was introduced in 2013 as it improves the training procedure by removing the hidden layer and approximating the objective (Mikolov et al., 2013.). Also in 2013, neural network models such as recurrent neural networks, convolutional neural networks, and recursive neural networks were implemented to perform NLP tasks.

Sequence-to-sequence was proposed in 2014 by (Sequence to sequence learning with neural networks, 2014) Sutskever et al.) Authors used an encoder neural network to process a sentence symbol by symbol and compress it into a vector representation. The principle of attention was proposed in 2015 by (Bahdanau 2014.) that considers one of the innovations in neural machine translation (NMT). Large pre-trained language models are without a doubt the most recent major innovation in the world of NLP. Pre-trained language model embeddings can be used as features in a target model or can be fine-tuned on target task data. The ability of these pre-trained language models to learn word representations from large unannotated text corpora is particularly effective for low-resources languages where labeled data is hard to obtain (Louis, Antoine, 2020).

2.3.2 Natural Language Processing Tasks.

NLP is used in almost everyday application or tool, NLP is utilized and modified in accordance with human need, the following are a Common NLP tasks and techniques:

- *Named Entity Recognition*

Named Entity Recognition (NER) is used to mark and identify important information in a given text, for example named entities. NER is implemented in order to mark and define key information in a text, also used to reorder unstructured data to finally obtain key words, sentences and entities. NER helps to define characters that exist in a book, or a city name in an article, NER is also able to categorize data. In case a domain-specific key-words are needed to be extracted from data, NER is considered difficult but yet effective task to be used, difficult due to the lack of availability of domain-specific words in a common corpus, but effective due to the ability of training modern models to understand domain-specific jargon. This makes named entity recognition a

powerful NLP task that can be domain-specific and obtain high accuracy results (Yadav & Bethard, 2019).

(Dai et al., 2019) used named entity recognition to extract medical information from Chinese electronic health records. BiLSTM neural network approach was used for this task with two pre-trained language models: Word2Vec and BERT. To train document representations, (Jin et al., 2018) used an internal medical natural language processing service to perform named entity extraction and detection from clinical notes, and then, composed the selected entities into a new text corpus.

- *Classification*

Classification is the task of labeling an entity as part of a specific group. For example, the decision if a tweet is either a hate speech or not. Classification is an important NLP task since it depends on extracting semantic meanings from a text or a sequence of words. Many NLP tasks provide linguistic analysis, text labeling and categorization are considered classification.

(Budiharto & Meiliana 2018) used sentiment analysis to predict the political elections from posts and comments on social media such as Twitter and Facebook. Natural language processing of electronic health records was used by (Gkotsis et al., 2017) to study the mental health conditions on unstructured social media data.

- *Relation Extraction*

Relation Extract is concerned with detecting semantic relationship within text body where entities are related. For example, identifying the relationship between symptoms and a certain disease or person and organization. Relation extraction is an important

NLP task, the outcome is crucial where identifying entities relation to each other is often lost when dealing large corpora or a specific domain text.

(Bekouli et al. 2018) introduced a joint neural model to perform relation extraction and entity recognition simultaneously without the use of any manually extracted features or any external tool. Relation extraction task used to determine multiple relations for each entity. Their work was based on different datasets and languages.

- *Question Answering*

Question answering is the steps taken to construct a model that can provide answers for questions, the form of answers is queries from a database. Many giant High-tech companies uses different models that perform queries on their databases, Google, Amazon, and Apple have gigantic database that contain anything mankind ever think about. Question answering is considered favored since it provides customer service applications in an excellent manner with high accuracy.

(McCann et al. 2018) introduced a Natural Language Decathlon model which consists of a question-answering format for a suite of ten NLP tasks. In 2020, (Khashabi et al. 2020) reformulate many QA tasks by fine-tuning seq2seq-based pre-trained models. Authors presented a UNIFIEDQA model for the purpose of bringing unification across four common QA formats.

- *Information Extraction*

Information Extraction (IE) is used to process unstructured data in order to produce structured data without the intervention of humans. This facilitates the extraction of important features from a given data such as entities relationships and producing attributes describing these extracted entities. IE utilizes NLP tasks fully to produce structured information readable and meaningful for computers to aid further process

such as storing data, processing, etc (Sarawagi et al., 2008; Wimalasuriya and Dou, 2010).

IE systems input of unstructured data can be of different types such as videos, images, and audio. And as prementioned IE systems produce meaningful information that can be used by computers and to achieve this, three effective approaches are used:

A. Pattern matching: Based on regular expressions, when a match is found for a given sequence of tokens. The result is an extracted matched text correlated with a specific entity. The downside of this approach is defining entities, since there will be text that cannot be correlated to an entity.

B. Gazetteer-based approach: Like the previous approach where entities have to be matched, but with the help of an external knowledge source such as pre-defined list. Matching between text data and entities is more accurate and efficient since they provide multiple names for the same entity. The downside of this approach is having to build a huge gazetteer and to make it accurate to achieve more accurate results.

C. Machine Learning-based approach: Machine Learning algorithms learn how to extract entities by training on a given set of examples, these examples are called “ground truth.” The training data (ground truth) is processed by annotation. So, we have entities with annotation and ML models uses POS tagging, word position, and other methods to produce a meaningful data annotation.

2.3.3 Word Embedding

As prementioned in section 2.3.2, NLP tasks produce structured information readable and meaning full for computers. Word Embedding is one of the most effective methods that NLP tasks use. Word embedding is simply a representation of words formatted as a vector.

Each word is translated to a binary value of 0 or 1, which is known as one-hot encoding. This allows us to find the presence of a word in a given sentence, where input data containing sentences usually has a repeated presence of the same word. By this we are informed about word presence or not in a document. These words are unsupervised learned word representation in a form of a vector, and these vectors helps to define semantic similarity, thus provide accurate results when trying to find certain behavior in a text such as similar words (Mandelbaum & Shalev, 2016).

The following two sentences are examples of how words are represented in word embeddings. For these two examples, a one-hot-encoded vector will be created. The vector is able to represent the words in these sentences in quantifiable way by encoding each word feature, and each feature will point the location of word in a binary way.

Example 1: A penny saved is a penny earned

Example 2: A penny for your thoughts

From the two sentences, a vocabulary will be created that is unrepeated and unique words only. “A penny, saved, is, earned, for, your, thoughts” and next is performing the one-hot-encoded process. The following table will show One-Hot-Encoded process, where “0” word does not exist in vocabulary, while “1” means word exist in the vocabulary.

Table 2. 2 Word Embedding One-Hot-Encoder

Example no.	A	penny	saved	is	earned	for	your	thoughts
Example 1	1	1	1	1	1	0	0	0
Example 2	1	1	0	0	0	1	1	1

This task allows the model to identify words presence in input data, but it does not show an important feature which is the relation of words with each other inside a given sentence such as cat – animal or kid – parents. This type of relation is called Bag of Words "BoW representation" (elaboration can be found in Section 2.3.4). BoW is used to gather and create word-count vector, this vector will have values of 0 or 1 depending on the appearance of the word in the vocabulary. An issue surfaced when using a large amount of training data called a spare matrix, which is basically a matrix where most of its vector elements are zeroes. This led to losing the relationship and meaning within word representation and also words order, thus the word representation does not contain any semantic relationship.

To overcome losing word order and semantic relationship, the principle of word-vector is used. Word-vector keeps words having similar meaning closer to each other. When a word is encoded immediately a vector space model is generated and modeled. This method was introduced by (Mikolov et al., 2013) and referred to it with word2vec.

- word2vec: When dealing with large loads of unstructured text data, word2vec is considered one of the most efficient ways for generating vector representations. One of the most beneficial perks of using vectors is its ability to use vector arithmetic to process and deal with word analogies. To demonstrate word2vec vectors, an example of algebraic vector based on word2vec method is presented below. The words relationship can be linear structured in the embedding space. Also, it became possible to propose distributional hypothesis, meaning it is

possible to characterize words by the words they are neighboring, this made it possible to define the context of the sentence. Figure 2.10 shows that the word "Queen" is more likely to be found around the word "King."

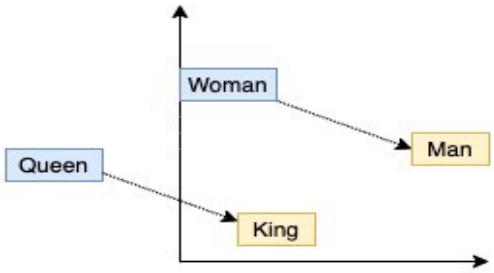
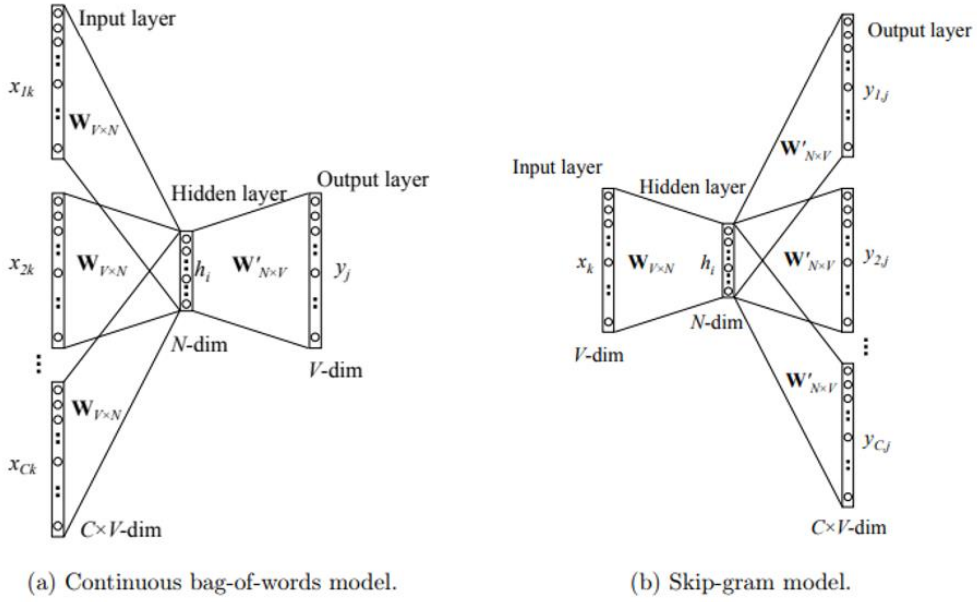


Figure 2. 10 Word Embedding Word2Vec (Mikolov et al., 2013)

Word2Vec uses two different language modeling methods in order to encode words into embeddings: Skip-gram model (SG) and continuous Bag-Of-Words (CBOW).

Skip-gram SG principle is to predict the context of a word from the given word, while CBOW predicts the probability of the targeted word within a window of given size. Figure 2.11 depicts Continuous bag-of-words (CBOW) and skip-gram (SG) architectures (Mikolov et al., 2013).



(a) Continuous bag-of-words model. (b) Skip-gram model.

Figure 2. 11: Skip-gram and continuous Bag-Of-Words structure (Mikolov et al., 2013)

When dealing with large size corpus word embeddings have several limitations including the order and relationship of words with each other. This made the Word2Vec approach unable to perform polysemy in an efficient way. Polysemy is simply defined as the same word that has various meanings. To demonstrate that, the word Turkey can be used within certain corpus to refer to a country, while in different context it can be used to refer to type of food served. Word2Vec will build single representation that connect Turkey with country as it builds the association of representations per word. In the case that Word2Vec is unable to handle polysemy, it will always refer Turkey to a country, and it will not be able to learn semantic or syntactic of words.

In addition to Word2Vec and FastText, there are other approaches that are used for obtaining dense word representations, such as ELMo and attention mechanism.

- FastText algorithm takes sub-word information in order to ensure that words with similar morphologies are allocated similar vectors. This algorithm is more appropriate in case of finding good representations for less frequent tokens in a certain corpus, also, based on n-grams this algorithm can help to understand vector of unseen words. Same as in Word2Vec, FastText algorithm uses CBOW and Skip-Gram structures.

ELMo algorithm is a way to find contextual word representations. In Word2Vec and FastText the ambiguous word such as “Bank” will get the same vector, regardless its meaning as a river bank or as a financial institution, but in ELMo, it will be certain that any ambiguous word will assign a vector based on its surrounding context. For attention mechanism that become popular through the last years, it is a tool that used for training language model where it contains large

transformer-based model such as BERT model (Bojanowski et al., 2016, Mandelbaum & Shalev, 2016)

2.3.4 Bag of Words

In natural language processing information retrieval, Bag of Words (BoW) model is considered one of the most used and effective to obtain object categorization task in a simplified way.

In BoW model, text is represented as unstructured form of data where words order, and grammar is ignored and considered irrelevant for achieving the task. BoW is concerned with a targeted word appearance in a document, not caring for its location, context, or grammar. BoW shines with feature extraction tasks, where the result is stored in a vector with identifiers. These Vectors identifiers represent the frequency of the words appearing in the data. Later these vectors which are rows in a matrix are used as features, which can be used to complete NLP downstream tasks (George K & Joseph, 2014).

2.3.5 Normalization

Normalization is considered as pre-task step, where its outcome helps other downstream tasks such as classification and NER. Normalization is responsible for transforming text to a more exact and specific format. An example is different dates format, where normalization takes different date formats and converts them to defined format such as DD/MM/YEAR. This step is important in specific-domain tasks such as medicine and in NLP, various scientific terms refer to and connected to terms used commonly. To demonstrate, the word salt and sodium chloride are the same, but a model could overlook that they are connected to the same entity.

2.4 Named-Entity Recognition

The major work for Named Entity Recognition (NER) has been in the Information Retrieval field. Although many methodological models have been used for Natural Language Processing, one of the most common and used models is Named Entity Recognition (NER). In 1995 and during the Message Understanding Conferences (MUC), the expression named entity was used. Named Entity Recognition and Classification (NERC) was the first term known, then it was changed to Named Entity Recognition (NER). Numerous worldwide conferences and shared projects have focused on NER, with prominent examples being the Conferences on Natural Language Learning and the Automatic Content Extraction research program. In NER research community and for practical purposes, there is a general consensus regarding the named entity, where the entity with no reference or non-rigid reference should be included without any restriction. Some annotated resources, for example, include entity groups for dates, developed words from names, and monetary expressions. (Ratinov & Roth, 2009., Yadav & Bethard, 2019)

The number of annotated resources is increasing over the years and becoming available in different languages and domains, and the data comes from different resources such as news articles, blogs, and social media. The early methods used for NER were mostly rule-based which provided high precision but unfortunately suffer from low recall values. With the advancement of methods, the supervised machine learning approach that needs an increasing amount of labeled data has become the main approach. Traditional supervised NER models were based on algorithms such as decision trees, support vector Machine (SVM), Hidden Markov Models (HMM), and Conditional Random Fields (CRF).

2.4.1 Label Encoding Scheme

The label encoding scheme is a method for encoding named entity labels, where positional information is given to classifiers regarding named entity tokens that will format the datasets. INSIDE-OUTSIDE-BEGINNING "IOB" labelling considers one of the prominent encoding schemes in NER as will be explained next.

- *IOB Labelling*

Ramshaw and Marcus in 1995 introduced IOB labeling scheme that has become now widely used for labeling named entities. In this scheme, the words are marked based on their location in relation to an entity, outside (O) or inside (I-), or (B-) at the beginning of the entity. This scheme has slight variations such as: excluding B-, we can use IO encoding scheme, IOB1 where B- is used when a named entity token is followed by a token belonging to the same entity, IOB2 where B- is used at the beginning of every named entity, IOBE scheme where the label E referred to end-token of the entity, IOBS scheme where the label S referred to single, unit-length entities, and finally there is IOBES. Research proved that training the models based on an IOBES-encoding scheme has improved the performance better than IOB2. (Christensen et al., 2010.)

Table 2.3 next shows a comparison of these schemes with an example from the NorNE corpus: "Islands president Olafur Ragnar Grimsson.". The annotation in NorNE has number of entity types, GPE-GPE Token stand for Organization, and PER Token stand for Person.

Token	IO	IOB1	IOB2	IOBE	IOBS	IOBES
Islands	I-GPE-ORG	I-GPE-ORG	B-GPE-ORG	B-GPE-ORG	S-GPE-ORG	S-GPE-ORG
president	O	O	O	O	O	O
Olafur	I-PER	B-PER	B-PER	B-PER	B-PER	B-PER
Ragnar	I-PER	I-PER	I-PER	I-PER	I-PER	I-PER
Grimsson	I-PER	I-PER	I-PER	E-PER	I-PER	E-PER
.	O	O	O	O	O	O

Table 2. 3 Encoding scheme comparison

2.4.2 Named Entity Recognition Approaches

NER is a word-by-word sequence labeling task, and it has traditional and modern models. The traditional models were rule-based on statistical models and modern models such as Conditional Random Fields consider baseline for sequence tagging.

- *Traditional NER approaches*

The extraction of feature set in traditional NER systems is done by extracting the target word and its surrounding context. Examples of the features being extracted are prefixes, capitalization patterns, syntactic chunk tags, suffixes, and word shapes. Word shapes indicate how the word is structured which is important when it comes to encoding, for instance, Olafur will be on the shape of Xxxxxx, and president will be xxxxxxxxx, due to capital letter in the word. When the features are extracted within a context window, it is considered as a local feature, and when it is extracted outside the window or from knowledge sources external to the document itself it is considered as a non-local. Based on (Ratinov & Roth, 2009), there are many non-local features and external sources containing context aggregation features, gazetteers, extended prediction histories, and unlabeled data.

Unlabeled text is used as a word class model which is an unsupervised approach that looks like vector space models for distributional similarity. In unlabeled dataset, the data will be grouped together hierarchically to generate a set of class-based features, where the words that have the same contexts will be assigned to the same groups. The Gazetteers-based approach considers large knowledge bases containing known named entities such as locations and person names.

Context aggregation features where the tokens in contexts are aggregated together forming identical labeling of tokens within the same contexts, and these features can be

extracted manually or defined manually. In extended prediction history where the labels that have been used for each word will be stored. In summary, the traditional NER models use mostly external knowledge resources and handcrafted features. All local and non-local features in the traditional models will be collected to form a single feature vector that will be passed to the learning model.

- *Modern NER approaches*

NER traditional models proved their ability but have a few drawbacks such as these models consume time when developing language-specific resources, also the space of feature extracted is of high dimension, and in case of non-linear classification problems the linear models are challenging.

The new approach for NER uses Artificial Neural Network (ANN) with few language-specific resources and few feature engineering. In this section, we will talk about the fundamentals of feed-forward networks, Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), training Neural Networks, and the recent trends revolving around the design of sequence labeling systems.

- *Feed-Forward Neural Network*

Neural Networks are learning models with numbers of neurons. Each neuron is a computational unit with scalar inputs and outputs and a corresponding weight. Each unit will multiply the elements with their weights and then sum their values. The result will be applied to an activation function (Goldberg, 2017). The architecture of a feedforward network shown in figure 2.12 is where a number of neurons are connected together forming a network with a computational flow that starts from an input layer, then a number of hidden layers, and ends with an output layer.

The mathematical representation of the feedforward neural network is as shown in equations next. For layer number i, $W[i]$ is the weight matrix, $b[i]$ is the bias, $z[i]$ is the combination of weights and biases, this combination will be applied to an activation function represented by $g[i]()$, then the output will be produced $a[i]$, where the final output will be $\hat{y}$, (Huang et al., 2019).

$$
\begin{aligned}
a^{[0]} &= x \\
z^{[1]} &= W^{[1]}a^{[0]} + b^{[0]} \\
a^{[1]} &= g^{[1]}(z^{[1]}) \\
z^{[2]} &= W^{[2]}a^{[1]} + b^{[2]} \\
a^{[2]} &= g^{[2]}(z^{[2]}) \\
\hat{y} &= a^{[2]}
\end{aligned}
\tag{2.7}
$$

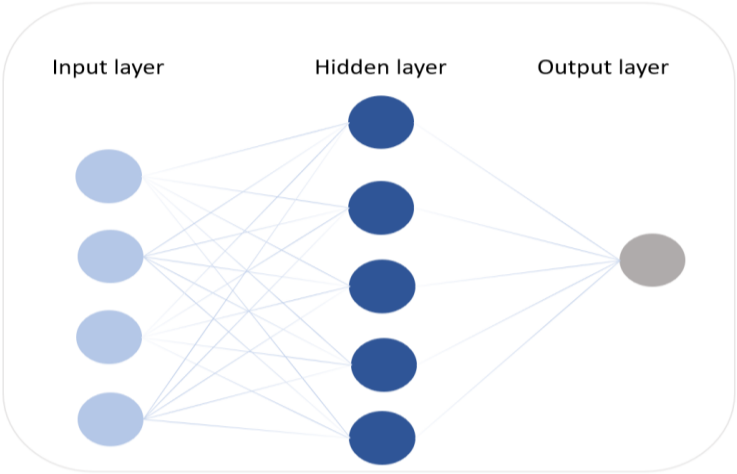


Figure 2. 12: Feedforward Neural Network structure *(Huang et al., 2019)*

- *Convolutional Neural Network*

Convolutional Neural Network (CNN) is a neural network that is designed to detect informative and local data within a larger structure and merge these in order to classify the input. CNN is used widely for image processing and has currently proven its ability in various NLP tasks. CNN as shown in figure 2.13 is composed of several stacked

layers arranged in three main components, starting with the convolutional layer, pooling layer, and fully connected layer. (Huang et al., 2019).

In convolutional layer, there is a filter that is applied to each word window in an input sentence, that will generate a vector with all the central properties of the words. Then, we have the pooling layer, which combines all of the previous vectors from the convolutional layer into a single vector. The pooling layer takes different operations to work, such as max pooling, average pooling, and sum pooling. The resultant vector will then pass to fully connected layers in order to make the final classification of the input.

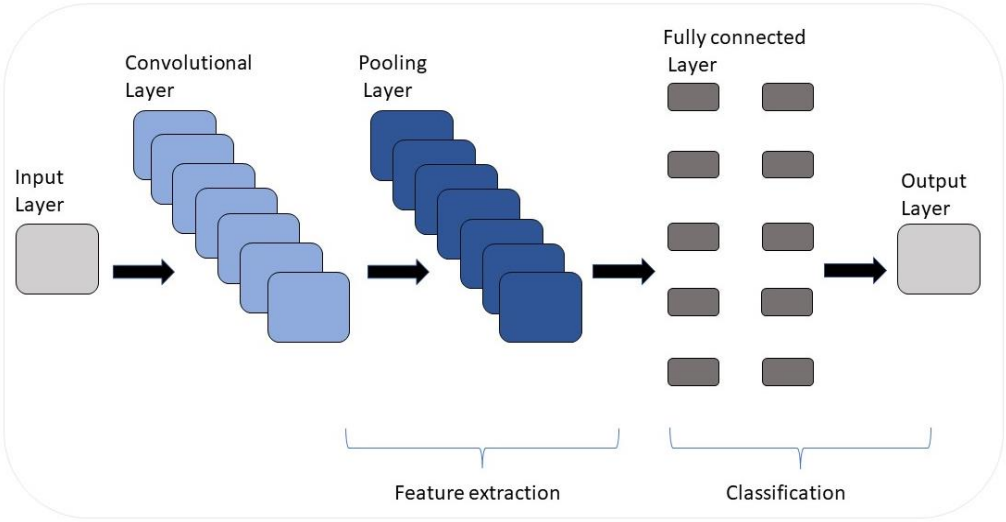


Figure 2. 13 Convolution Neural Network structure (Huang et al., 2019)

- *Recurrent Neural Network*

Recurrent Neural Network (RNN) is an artificial neural network that processes data based on time sequences such as natural language processing and speech recognition. In convolutional structures, the order of the input will not be stored beyond the range of the sliding windows, and so, such structures will face difficulties in dealing with long-distance relationships and dependencies between words in larger sequences. Recurrent neural networks are distinguished by the presence of memory gates as shown in figure

2.14 that gathers information to influence the instant input and output, as well as the presence of the weight parameter within each layer of the network (Goldberg, 2017).

In RNN, the internal loops allow us to pass the data between a series of cases, each corresponding to a specific input element. A new state will be generated out of the previous state vector and the current input vector. The formula of RNN as a function is shown next, (Goldberg, 2017).

$$RNN(x_{1:n};\ C_0) = y_{1:n}$$

$$y_i = O_{C_i}$$
$$C_i = R(C_{i-1}, x_i) \tag{2.8}$$

$x_{1:n}$ representing the input sequence, C_i is the state vector for a step i in the input. The recurrence formula is R, that takes the previous state vector C_{i-1} and the input vector x_i. O: is the function that maps the state vector C_i to an output vector y_i. The initial vector is C_0 that often chosen to be zero. In feedforward neural network, each layer has weight and bias, but in RNN, they share parameters for all states in the network.

The simple instantiation of RNN is known as Elman Network or the Simple-RNN, the states are defined as follows, where the current input x_i and the previous state C_{i-1} are linearly transformed, and the sum of these will be fed to a non-linear activation function g (). (Elman, J. L. 1990)

$$C_i = R(x_i, C_{i-1}) = g(C_{i-1}W^C + C_iW^x + b\ )$$

$$y_i = O(C_i) = C_i \tag{2.9}$$

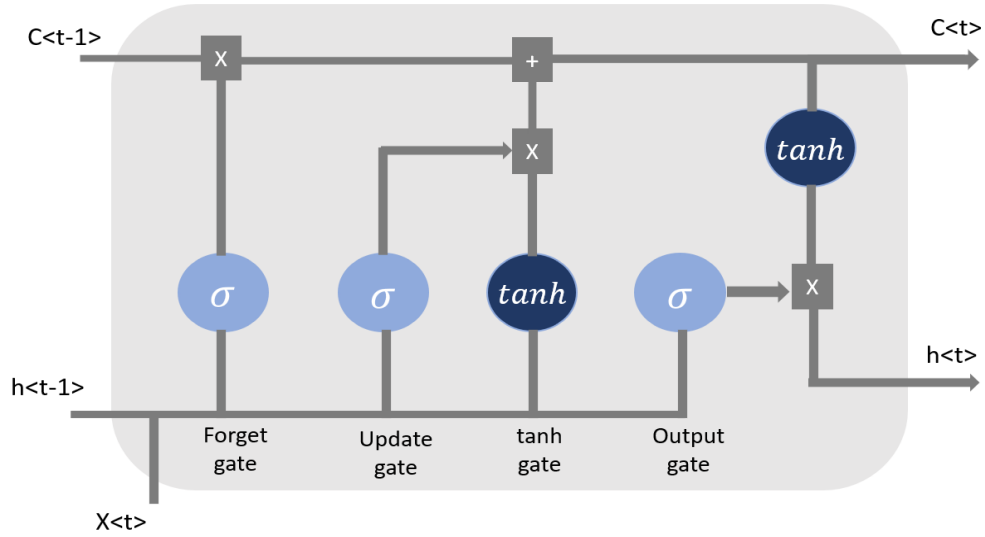


Figure 2. 14 Long Short-Term Memory structure (Hochreiter & Schmidhuber, 1997).

2.4.3 Neural Network Training Functions

- *Activation function*

The activation function in neural network maps the input to the output and can activate a neuron based on the network input value and a threshold. The input is applied to a threshold value to determine whether or not the neuron should be fired (Collobert et al., 2011;. Nwankpa et al., 2018). A brief overview of the most commonly used activation functions is as follows.

A. Sigmoid: Used for the binary classification at the output layer and can convert the real numbers into a probability that is easy to interpret. Equation (2) is the mathematical representation of a sigmoid function, where the output is a value between 0 and 1. (Nwankpa et al., 2018)

$$Sigmoid = \frac{1}{1+e^{-x}} \tag{2.10}$$

B. tanh: This function resembles the sigmoid function, where the input is real values, and the output values lie between -1 and 1. The mathematical representation of tanh function is in equation next:

$$tanh = \frac{e^{2x}-1}{e^{2x}+1} \tag{2.11}$$

C. ReLU: ReLU is used where it can improve the training when several hidden layers are used, also is able to decrease computations and improve performance while training. The mathematical representation of ReLU is in equation next, and the output from this function is 0 if it got a negative value, and the result will be the same if it got a positive value. (Nwankpa et al., 2018)

$$ReLU = max(0, X) \tag{2.12}$$

D. SoftMax function: Known as a normalized exponential function that converts a vector of numbers into a vector of probabilities where the sum of the probability's distribution is 1. The mathematical representation of Softmax is in equation next. (Nwankpa et al., 2018)

$$Softmax(x_i) = \frac{e^{X_i}}{\sum_j^n e^{X_j}} \tag{2.13}$$

Sigmoid, tanh, and ReLU activation function are applicable for hidden layers. SoftMax activation function is used for the classification task at the output layer.

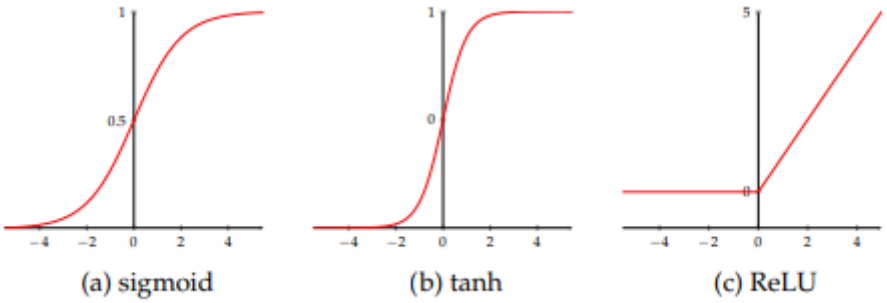


Figure 2. 15: Activation function comparison (Nwankpa et al., 2018)

- *Optimization function*

This function is responsible for changing model parameters such as weights and learning rate in order to get the most accurate model (Cho et al., 2014). There are many optimization functions, such as:

A. Stochastic Gradient Descent (SGD): This optimizer is used to determine which parameters best match the actual and predicted outputs, where it uses the learning rate to update the parameters based on its training sample.
B. Root Mean Square Propagation Optimizer (RMSProp): This optimizer is responsible for step size, where it will decrease the size in large gradients to avoid exploding and increase the size for small gradients to avoid vanishing.
C. Adaptive Moment Estimation (Adam): This optimizer is a replacement for stochastic gradient descent where it is used to reduce the loss in case of a classification problem. There are parameters in this optimizer that make it the fastest convergence optimizer, which are: the learning rate, a weighted average of squared gradient, and a weighted average of gradient.

- *Batch size*

Batch size represents the number of datasets included in one batch. Datasets used for training can be divided many times, depending on the testing approach, each dataset can be split to many batches representing the number of iterations. Iteration represents the number of batches needed to complete one Epoch.

- *Epoch*

One Epoch is when the whole dataset is passed in and out of a neural network. Epoch is used to improve accuracy and decrease Loss by increasing Epochs. Meaning that

training dataset is repeatedly and continuously fed into the neural network, thus helping to improve the weights.

- *Learning Rate*

When model Weights are being improved and updated, there is a need to reflect these changes on the model. Learning rate help with this, learning rate is a tricky value to be defined, if it is small its effect appears on the training process where it could be stuck or extend its training time, and if it is large it could result in unstable training process and affecting the weights setting.

2.5 Feature Representations And Unsupervised Pretraining

Extracting feature vectors encoding data is done during the training of sequence labeling system such as abstract word shapes. For One-Hot encoding, the result of categorical data will be in the form of a feature vector that has unique dimensions for each possible feature. As a result, this will lead to a feature space that is sparse and high-dimensional, which will cause a high computational cost, and in some NLP tasks, feature extraction takes longer than the parsing itself (Mikolov et al., 2013).

Traditional One-Hot encoder could have thousands of feature dimensions which will cause the "curse of dimensionality", and this is solved with the emergence of neural NLP as it uses dense feature vectors where hundreds of feature dimensions are used. In neural NLP, the core features that are passed to the neural structure are represented as vectors embedded into dense, n-dimensional space. A word embedding layer could be employed before the neural structure, as it will act as a look-up table for our dense word representations. And then, we can check if the input tokens are in the embedding layer during the processing of the input sequence, as the representations will be passed to the next layer in the neural network.

The word embedding layer is initialized randomly and it will be optimized as other networks during the training process. In unsupervised approaches where a big amount of unlabeled data is used to support NER models. Such an approach can be implemented in neural networks to guarantee that we have word representations for a certain task before starting the training process. Based on contexts, a representation of the words and unlabeled corpus is found in a raw during unsupervised pre-training. Words with similar meanings will have similar vectors since they are frequently seen in the same contexts. The representations learned from an unlabeled resource should be same as those learned from labelled data, and this will assist the models in terms of generalization capability. (Bojanowski et al., 2016)

Based on the task, the dense feature vectors are combined. For example, taking the average of all word vectors in the input sentence if we have classification problem, then, forwarding the averaged vectors into feedforward network. In terms of utility, there are many ways to generate the utility of dense vectors such as Word2Vec and fastText, mentioned in 2.3.3 (Mandelbaum & Shalev, 2016).

CHAPTER THREE
RESEARCH METHODOLOGY

3.1 Introduction

This chapter describes the methodology that is used in order to build a natural language processing model using AraBERT model processed on a new medical Arabic dataset.

3.1.1 Domain-specific BERT-based Models

A great attention has been drawn to the idea of a specific-domain models uses and adopt state-of-the-art language models to perform different tasks. The following will mention Models used English as language to understand and process.

Work done by Author (Huang et al., 2019) where a pre-trained BERT model was used on clinical notes. The steps were taken in one of their experiments where word similarity tasks were used to perform a comparison between ClinicalBERT and popular word embedding model, and they found that ClinicalBERT showed high correlation on intended medical concepts. As for NLP tasks the authors went for different approach where they fine-tuned their ClinicalBERT to perform clinical prediction and compared the result with two different models, these models used bag-of-words and the other used LSTM model with Word2Vec embeddings. The result was ClinicalBERT outperformed other models in prediction tasks.

Likewise, (Lee et al., 2021) with BioBERT. A BERT model was pre-trained using large biomedical corpus 5.5 times larger than used to pre-train BERT. With Fine-tuning to perform text mining on biomedical dataset, Question Answering, Named Entity Recognition and Relation Extraction tasks. Same as (Huang et al., 2019), word

BioBERT outperformed BERT and older state-of-the-art models (Lee et al., 2021) used different biomedical datasets.

In 2019 SciBERT was released by (Beltagy et al., 2019). Authors pre-trained BERT using large corpus related to scientific text. Their work was concerned with investigating the outcome of fine-tuning a pre-trained model to perform downstream tasks against pre-trained task-specific models. They evaluated various NLP tasks such as text classification and sequence labeling and concluded that fine-tuning performs better than using what he called feature-based approach.

In short, their work concluded that fine-tuned SciBERT outperform fine-tuned BERT. Also, another added contribution was when taken SciBERT and whether pre-train from scratch with domain vocabulary or start SciBERT with BERT weights on a certain downstream task, the two approaches will perform almost evenly and the difference is almost negligible, and the recommendation is to go with pre-training on specific corpus.

3.1.2 AraBERT

Arabic is considered a morphologically rich language, and low with resources in comparison with other languages especially English which made using Arabic in NLP tasks very challenging and almost undoable.

Due to modern Machine Learning models based on Transformers such as BERT and it is proven to be very efficient with language understanding. AraBERT creators pre-trained BERT for Arabic language and managed to achieve results similar to what BERT do with English. AraBERT managed to achieve state-of-the-art results on NLP tasks. Authors of AraBERT made their model available publicly on GitHub “hoping to encourage research and applications for Arabic NLP” (Antoun et al., 2020)

- *AraBERT Methodology*

Based on BERT base model (Devlin et al., 2018) which is proven to be a reference point for most of state-of-the-art results in different NLP tasks. BERT-base configuration has 12 encoder blocks, 768 inner dimension, 12 attention heads, 512 max sequence length, also a staggering 110 million Parameter. Authors developed Arabic language representation model AraBERT based on BERT-base, they described in their work pre-training setup and the pre-training dataset for AraBERT.

1. Pre-training Setup: Same as original BERT pre-training process, AraBERT authors used Masked Language Modeling (MLM) task where they made 15% of the N input tokens used for replacement. Out of 15% tokens they replaced 80% with [MASK] token, and 10% randomly tokens and the last 10% with original tokens. This forced the model to predict an entire word rather than guessing part of the word, then they used Next Sentence Prediction (NSP) task to help the model understand relationship between sentences.
2. Pre-training Dataset: The original BERT trained on 3.3B token which was extracted from English Wikipedia and the Book Corpus (Zhu et al., 2015). As for Arabic language, Arabic Wikipedia is very small compared to English Wikipedia, this forced AraBERT authors to find other sources such as (El-Khair, 2016) which is a modern corpus contain over than 5 million articles (1.5 billion word). And the main second source was (Zeroual et al., 2019), which is an open-source international Arabic News Corpus contain 3.5 million articles (almost 1Billion word). The final size of AraBERT pre-training dataset after cleaning the duplicate and extra cleaning process was 70 million sentences, in size was almost 24 Gigabyte. This dataset originated from corpus and contained a wide

range of topics which will be beneficial to different NLP tasks (Abdelali et al., 2020., Antoun et al., 2020).

3.2 Methodology Flowchart

Figure 3.1 depicts the methodology flowchart of our model based on Arabic dataset.

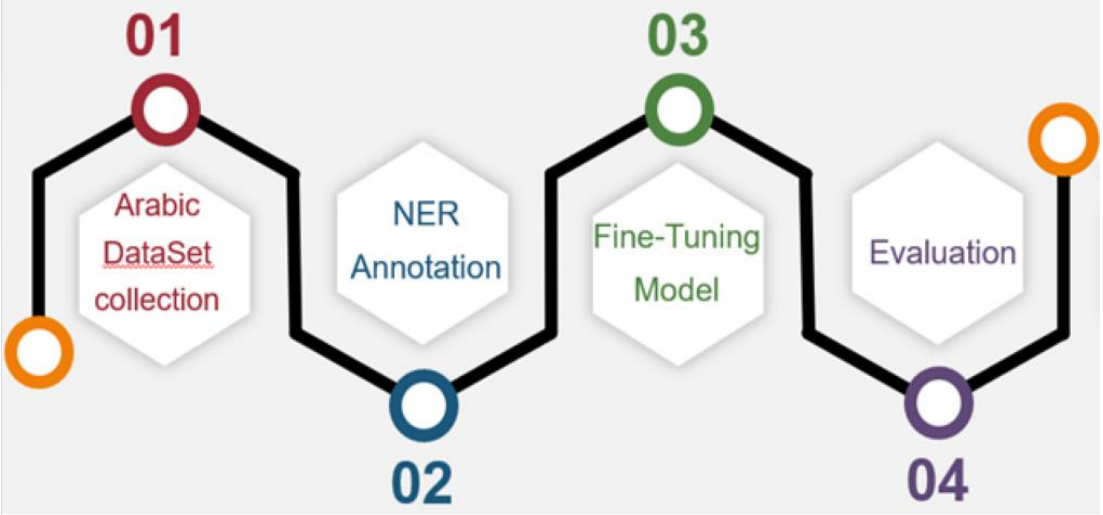


Figure 3. 1 Methodology flowchart

The steps of the proposed methodology are elaborated as follows:

3.2.1 Collect Arabic Dataset

The first step we did in our research is to find an accurate medical Arabic dataset which was not successful task, so manually we obtained a new one by gathering data from trusted Arabic medical sites (“دليل الامراض الشائعة - ويب طب”), (مصطلحات طبية | الطبي.” الطبي),“ our dataset contained professional diagnostic reports related to 200 Blood disease and 40 diagnose reports related to different type of disease.

At this stage all data gathered needs to be prepared before being fed to the model, this is because gathered data has different formats and inconsistencies especially with

Arabic language, the following bullet points are steps taken to prepare data for Feature selection step “annotation” which will be mentioned later.

- Different formats:
    - The reports were reshaped to be as separated paragraphs to facilitate annotation and cleaning diacritical marks.
    - Length and words count are taken into consideration to preserve consistency of size and words count across all paragraphs.
- Data cleaning: dataset now has consistency in form. Now we treat each paragraph as individual case and perform the following:
    - Removing diacritical marks to avoid misunderstanding and ambiguity. And avoiding any dialectal variations by using only classical Arabic.
    - Removing unneeded Punctuation marks such as Question mark, exclamation mark, comma`s and colon`s and quotations, hyphen and dash, unneeded numbers and letters, white-spaces, and English words.
- Prepare data for Feature Selection “annotation” and tokenization:
    - Testing and verifying of diagnostic reports quality and relevance to blood diseases by specialist working in this field.

The table below has a summary of words and sentences that later would be processed.

Table 3.1: Summary of the dataset

Sources	Class	Number of Words	Number of Sentences	word Annotated (B-disease)
مصطلحات طبية | الطبي.” الطبي دليل الامراض الشائعة - ويب طب	Blood disease	9940	400	1826

The below Figure 3.2 exhibits an example of a blood disease diagnostic report, which later will be processed using annotation.

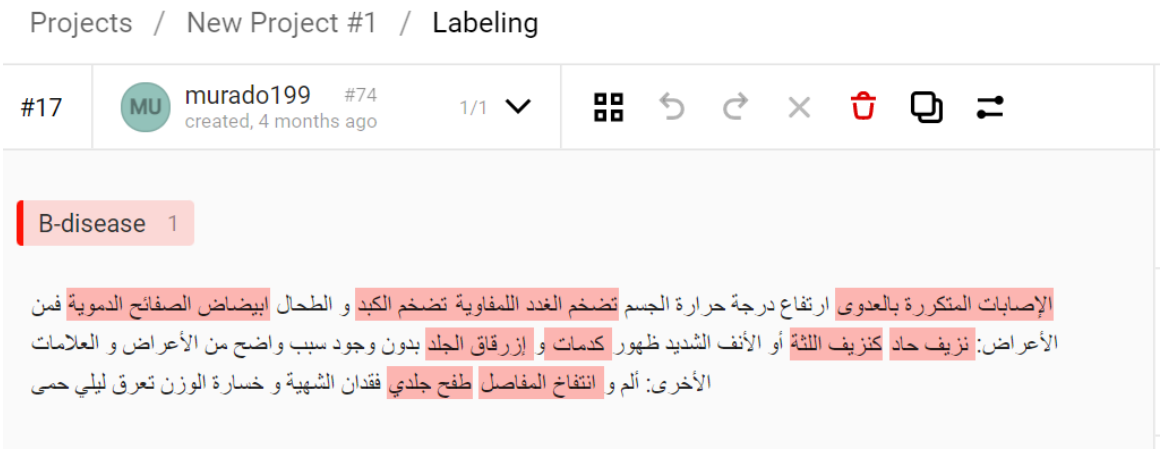


Figure 3. 2 Example of a blood disease diagnostic report

3.2.2 NER Annotation

After careful selection and gathering of the dataset, preprocessing step comes. This step is a crucial step before training the model due to its effect on the effectiveness of the model. In this step, we have to put the data in a form where it can be easy for the model to understand the data and by this, we refer to the “Tagging process.” The tagging process we used in our research is called “BIO Tagging.” BIO also refers to (IOB) which is a label encoding scheme mentioned in 2.4.2. The words are marked based on their location in relation to an entity, outside (O) or inside (I-), or (B-) at the beginning of the entity.

In our work we used an open source labeling tool used for labeling different data formats. We loaded the pre-processed text and started annotation and generating tags related to Blood disease. Table 3.1 shows the result of labeling stage.

Figure 3.3 shows an example of an annotated text. which later will be used to perform training process. Two formats produced, json and CONLL2003.

```
10271  النزيف -X- _ O
10272  فقدان -X- _ O
10273  الوزن -X- _ O
10274  والشعور -X- _ O
10275  بالتعب -X- _ O
10276  والإرهاق -X- _ O
10277  بشكل -X- _ O
10278  دائم -X- _ O
10279  فقر -X- _ O
10280  الدم -X- _ O
10281  وانخفاض -X- _ B-B-disease
10282  ضغط -X- _ I-B-disease
10283  الدم -X- _ I-B-disease
10284  تضخم -X- _ B-B-disease
10285  العقد -X- _ I-B-disease
10286  الليمفاوية -X- _ I-B-disease
10287  ما -X- _ O
10288  هو -X- _ O
10289  ابيضاض -X- _ O
10290  الخلايا -X- _ O
10291  البدينة -X- _ O
10292  شحوب -X- _ B-B-disease
10293  في -X- _ I-B-disease
10294  لون -X- _ I-B-disease
10295  الجلد -X- _ I-B-disease
10296  تعب -X- _ O
10297  وضعف -X- _ B-B-disease
10298  نزيف -X- _ I-B-disease
10299  متكرر -X- _ I-B-disease
10300  الإصابة -X- _ O
10301  بالعدوى -X- _ B-B-disease
10302  بشكل -X- _ I-B-disease
10303  متكرر -X- _ I-B-disease
10304
```

Figure 3. 3 Annotated Text

3.2.3 Fine-Tunning model

The most important step in our model is where we perform trials to find the best hyperparameters that provide our model with the highest accuracies and evaluation metrics. The hyper parameters we tried to tune to build the model were learning rate, epochs, and batch size. Experimental results of these hyper parameters are in chapter 4.

To elaborate more about Fine-Tuning, number of pre steps will be exhibited and explained:

- Calling our pre-trained model, along with its Tokenizer.

Here we instantiated BERT tokenizer since Arabert is built on BERT architecture, and along with it all weights and training procedure mentioned in 2.2.

- Pre-processing Dataset text using model Tokenizer.

In this step we reshape our data using Tokenizer, the output of this step is the same length Arabert uses 512 token which is the maximum token length BERT can accept, by this we avoided any issue when model start prediction process. Also, data is now Encoded where sequences have ID`s “numerical input representations” mentioned in 2.2.3.

The final output of this step is formed as Input_ids, attention_mask and Labels.

- Defining Training Parameters.

this step is very crucial, where the parameters that we use such as Optimization function which responsible to update learning parameters based on the output of the prediction, also Adam optimizer, which is helping to reduce loss in prediction, this goes for batch size, epoch and learning rate, mentioned in 2.4.3.

- Train and obtain a prediction.

In this stage we already reshaped our dataset, tokenized it to suit our mode, defined training parameters. To achieve an accurate prediction in this work dataset was divided into two parts, first part is used to train our model and has up to 80% of the whole data, the second part which 20% is used to perform a prediction.

The idea behind this is to test if the model with the defined parameters achieves successful prediction or not, keep in mind that 20% of the dataset is completely new to the model, which gives the model a high credibility.

3.2.4 Model Evaluation

To measure the effectiveness of the model, some evaluation metrices were used. We obtain accuracy, precision, recall, and f1-score and compare our results with another research. F1-score proves model learning capabilities, accuracy reflects the reliability to use the model, recall demonstrates the model's ability to predict labels, and precision shows. Section 2.2.6 contains detailed explanations and mathematical equations for each metric. A number of experiments were done to get the best metric as will be explained in chapter 4.

3.3 Summary

The details of the research methodology for developing a Blood Disease Prediction Model through Named Entity Recognition have been explained in this chapter. Starting with a flowchart that illustrates the general methodology steps. Then, a detailed explanation of each step is required to build the prediction model, beginning with collecting Arabic medical dataset from trusted medical sources. Following that, perform NER annotation based on BIO Tagging, and finally, fine-tune the model based on AraBERT in order to find the best hyperparameter for the model. The evaluation metrices were used to calculate accuracy, precision, recall, and f1-score in order to assess the model's effectiveness.

CHAPTER FOUR
FINE-TUNING RESULTS

4.1 Introduction.

This chapter provides detailed results for our disease prediction model, which was built based on AraBERT language prediction model. Our research started with collecting an Arabic dataset, then applied named entity recognition to find labels for our collected dataset. Following that, we performed a number of experiments to fine-tune the model to find the best metrics based on learning rate, batch size and epochs. We evaluated our model based on the following metrics: accuracy, precision, recall, and F1-score.

4.2 Model Setup

The experiments in our research were conducted by a personal computer. The software and hardware specifications are explained next.

4.2.1 Hardware Setup

We have built an evaluation platform on Dell Inspiron 15 3511 with Intel(R) Core (TM) i7-1165G7 @ 2.80GHz and 8.00 GB RAM.

4.2.2 Software Setup

The disease prediction model was implemented based on Python programming language, in addition to different libraries, as shown next in table 4.1

Table 4. 1: Software specification

LIBRARY	VERSION
Python	3.9.6
AraBERT	arabertv02
Matplotlib	3.5.1
Scikit-learn	1.0.2
Torch	1.8.1

Numpy	1.22.3
transformers	4.20.0
Pandas	1.3.5

4.3 Experimental Scenarios

In our research, after collecting the Arabic dataset, we performed experiments as illustrated in table 4.2 next.

Table 4. 2: Experimental Scenario

Dataset	Type of Labels	Learning Rate	Epoch	Batch Size
Blood Disease	>> Disease Symptoms: **B-B-disease** >>Non-disease Symptoms: **O**	2×10^{-5}	6, 10, 15, 20	16, 32, 64
		3×10^{-5}		
		4×10^{-5}		
		5×10^{-5}		
		6×10^{-5}		

4.4 FINE-TUNING RESULTS

In this section, we will show the experimental results we got from fine-tuning the disease prediction model based on Arabic medical dataset.

After collecting our Arabic dataset from medical trusted resources, the dataset has been cleaned and organized into the form of sentences to be tagged based on BIO-tagging model to create labels and now the dataset is ready to be processed. In order to find the best model that can give the best ability to predict the existence of blood disease, we fine-tuned our model based on three parameters that affect the performance of the model and are adjusted during the training process, and these parameters are: learning rate, batch Size, and epochs.

4.4.1 Fine-tuning based on changing Batch Size

The first part of the experiments was based on changing the number of batch sizes and learning rate. We have tested the model based on five values of learning rate:2×10^{-5}

till $6 \times 10^{-5}$and three values of batch size: 16, 32, and 64 at epoch =6. The results are shown in table 4.3, 4.4, and 4.5.

Table 4. 3: Evaluation metrices when batch size=16 and epoch=6

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	88.26	86.92	62.73	72.86
3×10^{-5}	90.68	89.32	71.45	79.39
4×10^{-5}	92.60	91.27	78.00	84.12
5×10^{-5}	94.24	92.57	83.82	87.98
6×10^{-5}	94.75	94.12	84.36	**88.97**

As shown in table 4.3 above, for batch size=16 and epoch=6, increasing the value of the learning rate has improved the performance of the model where at 2×10^{-5} learning rate the values for f1-score and accuracy were 72.86% and 88.26%, and increasing the rate to 6×10^{-5} improved the performance, were the values of f1-score and accuracy become 88.97% and 94.75, respectively.

Table 4. 4: Evaluation metrices when batch size=32 and epoch=6

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	88.26	86.90	62.73	72.86
3×10^{-5}	90.68	89.32	71.45	79.39
4×10^{-5}	92.60	91.27	78.00	84.12
5×10^{-5}	94.24	92.57	83.82	87.98
6×10^{-5}	94.75	94.12	84.36	**88.97**

Table 4.4 and table 4.5 showed the same results. The performance of the model improved while improving the value of the learning rate. For the same values of learning rate and epoch, increasing batch size did not have any influence on the performance of the prediction model. As shown in three tables, we got same values of accuracy, precision, recall, and f1-score for different values of batch size, with the highest values 94.75%, 94.12%, 84.36%, and 88.97%, respectively at learning rate = 6×10^{-5}.

Table 4. 5: Evaluation metrices when batch size=64 and epoch=6

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	88.26	86.90	62.73	72.86
3×10^{-5}	90.68	89.32	71.45	79.39
4×10^{-5}	92.60	91.27	78.00	84.12
5×10^{-5}	94.24	92.57	83.82	87.98
6×10^{-5}	94.75	94.12	84.36	**88.97**

4.4.2 Fine-tuning based on changing Epoch

The second part of our testing was based on changing the number of epochs. We have tested the model based on five values of learning rate:2×10^{-5} till $6 \times 10^{-5}$and four values of epoch:6, 10, 15, and 20 at batch size = 16. Results when changing the number of epochs showed different effects as explained in tables 4.3, 4.6, 4.7, and 4.8.

Table 4. 6: Evaluation metrices when batch size=16 and epoch=10

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	93.42	87.18	86.55	86.86
3×10^{-5}	96.99	94.49	93.45	93.97
4×10^{-5}	98.08	96.35	96.00	96.17
5×10^{-5}	98.82	98.15	96.73	**97.44**
6×10^{-5}	98.41	97.42	96.18	96.80

Table 4.6 above shows the results obtained for different values of learning rate with batch size = 16 at 10 epochs. As in the previous results, increasing the value of the learning rate will increase the evaluation metrices which means improving the prediction of the model. From table above, the best value of f1-score achieved was 97.44% at learning rate = 5×10^{-5}, which is much better than the value achieved at lower number of epochs (epoch = 6) as in table 4.3 were the highest value was 88.97% at learning rate = 6×10^{-5}. Looking at other metrics, increasing number of epochs from 6 to 10 has given the model better ability to predict our blood disease symptoms. The values of accuracy, precision, and recall are also improved, at learning rate = 5×10^{-5},

the value of the accuracy raised from 94.24% to 98.82%, for the precision from 92.57% to 98.15%, and for recall: from 83.82% to 96.73%.

Table 4. 7: Evaluation metrices when batch size=16 and epoch=15

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	97.90	95.00	96.73	95.86
3×10^{-5}	99.10	97.84	98.55	98.19
4×10^{-5}	99.36	98.38	99.10	98.73
5×10^{-5}	99.59	98.53	99.82	**99.19**
6×10^{-5}	99.54	98.39	99.82	99.10

Table 4. 8: Evaluation metrices when batch size=16 and epoch=20

Learning rate	Accuracy	Precision	Recall	F1-score
2×10^{-5}	98.58	95.29	99.27	97.24
3×10^{-5}	99.50	98.56	99.45	99.00
4×10^{-5}	99.82	99.28	100	99.64
5×10^{-5}	99.91	99.82	99.82	**99.82**
6×10^{-5}	99.86	99.46	100	99.73

Same results obtained in table 4.7 and 4.8, as increasing learning rate with the same value of epoch and batch size will give the model better ability to predict. The results are also show that increasing number of epochs when training the model will have a great effect on its ability to predict as obtained from evaluation metrices. At number of epochs = 15, the highest values of accuracy and f1-score were 99.59% and 99.19% at learning rate = 5×10^{-5}, where the precision and recall got 98.53% and 99.82%.

4.5 Summary

Results have been mentioned in this chapter. Starting with an explanation about the hardware and software experimental setup for this research. Then, the scenarios we put to reach the best performance. Following that, the outcomes of fine-tuning our model through changing the hyperparameters values. The effect of changing learning rate at fixed value of batch size and epoch is investigated. Then, the effect of changing batch sizes and epochs is also tackled. Based on the evaluation metrices such as accuracy, precision, recall, and f1-score the performance was studied and investigated.

CHAPTER FIVE
DISCUSSION

The appearance of models that are based on transformers such as AraBERT made NLP tasks easier to tackle. Literature review show difficulties using NER task to perform tasks in Arabic due to lack of Datasets sources. AraBERT, a special variant of BERT that is trained to understand Arabic language. AraBERT will be used in a different context by altering and modifying the model to help us achieve a successful prediction from a specific-domain Arabic dataset

In this work we created Medical Dataset and used it to train Arabert to perform prediction. The step of annotation was made and entities where produced, this was not addressed before for medical datasets where all previous data sets are general.

After finishing prediction and all pre steps leading to final prediction, an improvement is possible on two sides, the Arabert model itself and Dataset.

As for the Model, the current state of Arabert is giving state-of-the-art results, but it is depending heavily on the training and dataset, I suspect choosing different tokenization type could lead to a different result, The previous literature review (Abdul et al., 2021). (Abdelali et al., 2020.). (Antoun et al., 2020). related to Arabert shows Arabert capabilities in general domains.

Also, for Dataset. the prementioned literature review shows the lack of authenticated sources for specific domain datasets

The highest accuracy achieved during our research was batch size = 16, and epoch = 20. As illustrated in table 4.8, we acquired 99.91%, 99.82%, 99.82%, and 99.82%, for accuracy, precision, recall, and f1-score, respectively, and that match our objective to

find a model that is able to predict symptoms of disease based on Arabic dataset, this result answered our first and second research questions, where our modified model achieved a successful prediction using specific-domain Arabic dataset, and when fine-tuning is done correctly a higher accuracy achieved.

5.1 Benchmarking

To see the effectiveness of our model, we have to see the evaluation metrices resulted from different researchers.

Table 5.1 shows the results obtained from our model that used Arabic medical dataset based on AraBERT, compared to (Hammoud et al., 2020) that based on Arabic medical dataset based on BERT model, compared also to Hammoud et al., 2020 that used Arabic dataset based on AraBERT v2. The comparison was at learning rate = 2×10^{-5}, as both researchers used this value during fine-tuning process.

Table 5.1: Benchmarking based on Arabic dataset

Author	Model	F1-score
Hammoud et al., 2020	BERT	69.85 %
Hammoud et al., 2021	AraBERT v2	94.13 %
Our model	AraBERT v2	97.24 %

The results show that our model achieved the best f1-score based on only 2×10^{-5} learning rate with 97.24%, compared to other researches with 69.85% and 94.13%. This benchmark proved that having a specific-domain dataset improves prediction accuracy, since other models had a general dataset.

5.2 Summary

The end of this chapter was a comparison between our model and other models that used Arabic medical dataset and used new language prediction model. The best performance obtained from our testing and experiments was at learning rate = 5×10^{-5}, batch size = 16, and epoch =20, with 99.91%, 99.82%, 99.82%, and 99.82% for accuracy, precision, recall, and f1-score, respectively. The final results demonstrate the model's capability and robustness to predict blood disease symptoms.

CHAPTER SIX
CONCLUSIONS

This chapter will answer our Research question. Also discuss any challenges and limitations observed.

Question 1: Does AraBERT fine-tunning achieve a successful prediction using specific-domain Arabic dataset?

Yes, it does, according to the methodology used in this work, the result showed in 4.4.2 showed that when AraBERT was fine-tuned we achieved a successful prediction. we acquired 99.91%, 99.82%, 99.82%, and 99.82%, for accuracy, precision, recall, and f1-score, respectively.

Question 2: Does AraBERT fine-tunning achieve higher accuracy compared to another model that uses Arabic dataset?

Yes, it does, When AraBERT was manipulated through Fine-tuning our model still achieved a successful prediction. Section 4.4 shows all trials and tests done.

Question 3: Does creation of specific-domain dataset help the model to achieve successful and more accurate prediction?

Yes, it does, previous work from the literature review shows the poor results obtained when model used specific-domain dataset. while using a specific-domain dataset with proper fine-tuning will facilitate achieving accurate prediction.

Question 4: Can our model perform a prediction for another domain, e.g., criminal domain?

Based on literature review and other researches work, the answer of this question is more theoretically and is suggested as a future work. The current language models that understand Arabic are flourishing and great attention is withdrawn due to state-of-the-art results achieved. This led to having a gap in official and accurate datasets that can be used, where most of Arabic dataset set available are the result of individual researchers. In this thesis we created our specific medical dataset for the purpose of providing a prediction. A work done by researched from Poland to detect criminal Text on the web uses similar approach to the one used in this work, (Skórzewski et al., LREC 2022). Our approach can be used on different types of specific -domain dataset, for example having a dataset created from police reports and investigation results and processing them in the same manner to predict a certain behavior is doable, based on the prementioned work (Skórzewski et al., LREC 2022). BERT made This applicable due to knowledge transfer abilities, and the possibility to modify language model (AraBERT) in accordance with the result needed.

6.2 challenges and limitations

Challenges was faced during this work can be divided into the following:

- Software and Hardware

Most of the advanced machine learning tools can be found online, access to deploy python script is free, but it can be limited, this to make sure that service is available to anyone. Also, software skills are crucial and needed to take full advantage of the available ML Models.

- Data acquiring

One of the most difficult tasks was trying to get medical data from local hospitals. Due to sensitivity of data needed, this was time consuming and could be reason to delay the work needed, a work around was suggested is obtain data from trusted online-sources.

6.3 Future work

From a research perspective, Arabic is showing a great potential, especially with datasets, and it is gaining the attention of researchers. also, attention should be drawn to deal with Arabic dialects.

The Future work will include:

- Publishing a Paper regarding the approach used in this thesis. the collection method, and trying to obtain official data from public health ministry.
- Trying other Models, there are number of models that should be explored and tested.
- According to my research question “4”, research can be conduct to collect dataset created from police reports, and a Downstream can be used to answer a research question.

REFERENCE

Chowdhury, S. A., Abdelali, A., Darwish, K., Soon-Gyo, J., Salminen, J., & Jansen, B. J. (2020, December). Improving Arabic text categorization using transformer training diversification. In Proceedings of the fifth arabic natural language processing workshop (pp. 226-236).

Abdul-Mageed, M., Elmadany, A., & Nagoudi, E. M. B. (2020). ARBERT & MARBERT: deep bidirectional transformers for Arabic.

Altibbi.com, and @Altibbi. "مصطلحات طبية | الطبي.الطبي." altibbi.com/%D9%85%D8%B5%D8%B7%D9%84%D8%AD%D8%A7%D8%AA-%D8%B7%D8%A8%D9%8A%D8%A9. Accessed 10 Sept. 2022.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based Model for Arabic Language Understanding. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection, pages 9–15, Marseille, France. European Language Resource Association.

Beltagy, I., Lo, K., & Cohan, A. (2019). *SciBERT: A Pretrained Language Model for Scientific Text*. http://arxiv.org/abs/1903.10676

Bekoulis, G., Deleu, J., Demeester, T., & Develder, C. (2018). Joint entity recognition and relation extraction as a multi-head selection problem. Expert Systems with Applications, 114, 34-45.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, 135-146.

Budiharto, Widodo & Meiliana, Meiliana. (2018). Prediction and analysis of Indonesia Presidential election from Twitter using sentiment analysis. Journal of Big Data. 5. 10.1186/s40537-018-0164-1.

Cho, K., Merrienboer, B.V., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing*.

Christensen, J., Soderland, S., & Etzioni, O. (2010). *Semantic Role Labeling for Open Information Extraction*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. Journal of machine learning research, 12(ARTICLE), 2493-2537.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://github.com/tensorflow/tensor2tensor

George K, S., & Joseph, S. (2014). Text Classification by Augmenting Bag of Words (BOW) Representation with Co-occurrence Feature. *IOSR Journal of Computer Engineering*, *16*(1), 34–38. https://doi.org/10.9790/0661-16153438

Gkotsis, G., Oellrich, A., Velupillai, S. et al. Characterisation of mental health conditions in social media using Informed Deep Learning. Sci Rep 7, 45141 (2017).

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, X., Fan, X., Ying, J., & Chen, S. (2019). Emerging trends and research foci in gastrointestinal microbiome. *Journal of Translational Medicine*, *17*(1). https://doi.org/10.1186/s12967-019-1810-x

Jin, M., Bahadori, M. T., Colak, A., Bhatia, P., Celikkaya, B., Bhakta, R., ... & Kass-hout, T. (2018). Improving hospital mortality prediction with medical named entities and multimodal learning. arXiv preprint arXiv:1811.12276.

Khashabi, D., Min, S., Khot, T., Sabharwal, A., Tafjord, O., Clark, P., & Hajishirzi, H. (2020). Unifiedqa: Crossing format boundaries with a single qa system. arXiv preprint arXiv:2005.00700.

Lee, D. S., Fahey, D. W., Skowron, A., Allen, M. R., Burkhardt, U., Chen, Q., Doherty, S. J., Freeman, S., Forster, P. M., Fuglestvedt, J., Gettelman, A., de León, R. R., Lim, L. L., Lund, M. T., Millar, R. J., Owen, B., Penner, J. E., Pitari, G., Prather, M. J., … Wilcox, L. J. (2021). The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018. *Atmospheric Environment*, *244*. https://doi.org/10.1016/j.atmosenv.2020.117834

Louis, Antoine. (2020). NetBERT: A Pre-trained Language Representation Model for Computer Networking.. 10.13140/RG.2.2.27274.90564.

Mandelbaum, A., & Shalev, A. (2016). *Word Embeddings and Their Use In Sentence Classification Tasks*. http://arxiv.org/abs/1610.08229

McCann, B., Keskar, N. S., Xiong, C., & Socher, R. (2018). The natural language decathlon: Multitask learning as question answering. arXiv preprint arXiv:1806.08730.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. http://arxiv.org/abs/1310.4546

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. http://arxiv.org/abs/1811.03378

Ratinov, L., & Roth, D. (2009). *Design Challenges and Misconceptions in Named Entity Recognition * † ‡*. http://l2r.cs.uiuc.edu/

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14). MIT Press, Cambridge, MA, USA, 3104–3112.

van Barneveld, M., & Le-Khac, N.-A. (2016). *Performance Evaluation of a Natural Language Processing approach applied in White Collar crime investigation Adversarial Deep Learning and XAI for Cyber Security View Project Proposition*

et implémentation d'une technique de clustering à base de la théorie des jeux View project. https://www.researchgate.net/publication/307636386

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762

Wu, Y., Schuster, M., Chen, Z., Le, Q. v., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. http://arxiv.org/abs/1609.08144

Yadav, V., & Bethard, S. (2019). *A Survey on Recent Advances in Named Entity Recognition from Deep Learning models*. http://arxiv.org/abs/1910.11470

Z. Dai, X. Wang, P. Ni, Y. Li, G. Li and X. Bai, "Named Entity Recognition Using BERT BiLSTM CRF for Chinese Electronic Health Records," *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2019, pp. 1-5, doi: 10.1109/CISP-BMEI48845.2019.8965823.

"دليل الامراض الشائعة - ويب طب." دليل الامراض الشائعة - ويب طب, www.webteb.com/diseases. Accessed 10 Sept. 2022.

Jay Alammar. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Retrieved from http://jalammar.github.io/illustrated-bert, 2019.

Weiss, G., Goldberg, Y., & Yahav, E. (2017). Extracting automata from recurrent neural networks using queries and counterexamples. In: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, pp 5244–5253, http://proceedings.mlr.press/v80/weiss18a.html

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415, 2016.

Skórzewski, P., Pieniowski, M., & Demenko, G. (2022, June). Named Entity Recognition to Detect Criminal Texts on the Web. In Proceedings of the Thirteenth Language Resources and Evaluation Conference (pp. 6223-6231).

الملخص

أدى ظهور النموذج BERT " Bidirectional Encoder Representations from Transformer" الى إعطاء نتائج ثورية وغير مسبوقة في الأنظمة المتعلقة بمعالجات اللغة الطبيعية.

المراجعات الأدبية تظهر ان BERT قد تفوق وقام بإعطاء نتائج دقيقة عند استخدام بيانات متعلقة بتخصصات دقيقة، وبالذات عند عمل تعديل وموائمة للنظام حسب البيانات المطلوب عمل توقع او استنتاج منها.

قبل ظهور BERT التعامل مع بيانات من مجالات دقيقة يعتبر تحدي كبير لمعالجات اللغة الطبيعية، بسبب كون هذه البيانات تحتوي على مصطلحات عليمة وتقنية دقيقة. بهذه الرسالة سنقوم باستعمال النموذج AraBERT, وهو يعتبر نموذج مستنبط ومبني من النموذج BERT, حيث سيتم التغيير والتعديل ثم تدريب النموذج AraBERT على فهم بيانات طبية تحتوي اعراض مرض محدد، تم جمعها بشكل يدوي من اجل تنفيذ توقع ناجح عند سؤاله واختباره عن وجود هذا المرض او عدمه من بينات ذات علاقة واجابة أسئلة البحث الخاص بنا.

النتيجة النهائية أظهرت قدرة النموذج الذي تم تعديله وتدريبه من طرفنا AraBERT على فهم بيانات التي جمعها، واجراء توقع ناجح والحصول على نتائج ممتازة حيث تم استخدام نظام تقييم لإظهار دقة النتائج.