NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Greek-RoBERTas: Two Language Models for Greek

Myrto V. Iglezou

Supervisors: **Manolis Koubarakis,** Professor
Despina - Athanasia Pantazi, PhD Candidate

ATHENS

August 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Greek-RoBERTas: Δύο Γλωσσικά Μοντέλα για τα Ελληνικά

Μυρτώ Β. Ιγγλέζου

Επιβλέποντες: **Μανόλης Κουμπαράκης,** Καθηγητής
Δέσποινα – Αθανασία Πανταζή, Υποψήφια Διδάκτωρ

ΑΘΗΝΑ

Αύγουστος 2022

BSc THESIS

Greek-RoBERTas: Two Language Models for Greek

Myrto V. Iglezou

S.N.: 1115201700038

SUPERVISORS: **Manolis Koubarakis,** Professor
Despina - Athanasia Pantazi, PhD Candidate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Greek-RoBERTas: Δύο Γλωσσικά Μοντέλα για τα Ελληνικά

Μυρτώ Β. Ιγγλέζου

Α.Μ.: 1115201700038

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Μανόλης Κουμπαράκης,** Καθηγητής
Δέσποινα – Αθανασία Πανταζή, Υποψήφια Διδάκτωρ

ABSTRACT

BERT is one of the greatest contributions to the field of natural language processing, and many applications based on it are increasingly used. Every day more and more models are created based on this technology and its various variations. RoBERTa is one of them and is the main tool on which this work was based. We therefore present two models based on this architecture, trained on Greek language data. Examining their performance and comparing it with the existing model GREEK-BERT in the tasks of Part-of-Speech Tagging and Named Entity Recognition, we conclude that one of the ones we created has the best performance.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: RoBERTa, BERT, Deep Learning, Neural Networks, Natural Language Processing

ΠΕΡΙΛΗΨΗ

Το μοντέλο BERT αποτελεί μια από τις μεγαλύτερες συνεισφορές στον τομέα της επεξεργασίας φυσικής γλώσσας και πολλές εφαρμογές βασισμένες σε αυτό χρησιμοποιούνται όλο και περισσότερο. Κάθε μέρα δημιουργούνται όλο και περισσότερα μοντέλα βασισμένα σε αυτή την τεχνολογία και σε διάφορες παραλλαγές της. Το RoBERTa αποτελεί μία από αυτές και είναι και το κύριο εργαλείο στο οποίο βασίστηκε αυτή η πτυχιακή εργασία. Παρουσιάζουμε λοιπόν δύο μοντέλα βασισμένα σε αυτή την αρχιτεκτονική, εκπαιδευμένα σε ελληνικά δεδομένα. Εξετάζοντας την απόδοσή τους και συγκρίνοντάς τα με το ήδη υπάρχον μοντέλο GREEK-BERT σε δοκιμές Part-of-Speech Tagging και Named Entity Recognition, καταλήγουμε πως ένα από αυτά που δημιουργήσαμε έχει τις καλύτερες επιδόσεις.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Τεχνητή Νοημοσύνη

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: RoBERTa, BERT, Βαθιά Μάθηση, Νευρωνικά Δίκτυα, Επεξεργασία Φυσικής Γλώσσας

Ευχαριστώ φίλους και συγγενείς οι οποίοι, ο καθένας με τον δικό του τρόπο, με βοήθησαν να φτάσω μέχρι εδώ.

ACKNOWLEDGEMENTS

Firstly, I would like to thank Mr. Koubarakis, who through his lesson and the love he shows for the subject and the field of Artificial Intelligence in general, inspired me to explore this research area and finally do my thesis on a relevant topic. Secondly, I would like to sincerely thank Despina-Athanasia Pantazi, who was there to help me with any questions I had and to guide me through at any part of this process. They are both great collaborators and scientists and it was my pleasure to work with them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

A subfield of Artificial Intelligence called Natural Language Processing (NLP) [3] enables computers to comprehend, analyze, and use human languages. NLP enables machines and people to communicate using human language. Computers can read text, hear speech, and interpret it with the aid of NLP.

After BERT [6] model was introduced, only four years ago, the area of NLP witnessed a significant shift. The model managed to achieve cutting-edge results in a wide variety of challenging NLP tasks, including Question Answering, Natural Language Inference, and others. The primary technological advancement of BERT is the application of Transformer's [33] bidirectional training, a popular attention model, to language modeling. Transformer-based language models, acquire the characteristics of languages by unsupervised pretraining methods on massive amounts of text.

In order to improve the performance of BERT some researchers at Facebook tried to train the model longer, with larger batches and with more data and they did succeed in improving scores on downstream NLP tasks. They named the model RoBERTa [12] and this is the architecture we used in this certain research work.

The main target of this thesis is to develop a model based on the RoBERTa architecture and compare it to a BERT-based model in the same tasks. The team of Athens University of Economics and Business have already published a model called GREEK-BERT [10], which is trained on 29 GB of Greek text data and has achieved great performance in the tasks of Greek PoS tagging, Named Entity Recognition (NER), and Natural Language Inference (NLI). So, our challenge is to create a RoBERTa-based model and examine if it can perform better.

This thesis is divided in the following five chapters:

- In **chapter 2**, we provide some details of the theoritical background needed to fully understand the content of this thesis.
- In **chapter 3**, we analyze how the BERT and RoBERTa models work, we give a look into the GREEK-BERT model and we make some references to RoBERTa based models for other languages.
- In **chapter 4**, we introduce two models based on the architecture of RoBERTa trained on different combinations of Greek datasets.
- In **chapter 5**, we examine the tasks we evaluated our models with and the data we needed.
- In **chapter 6**, we present the results of this research and we make some conclusions based on it.

2. BACKGROUND

2.1 Machine Learning

Machine learning [17] (ML) is a subfield of artificial intelligence (AI) which focuses on optimizing algorithms in order to perform complex tasks in a way that is similar to how humans solve problems. Nowadays the approaches used in ML are widely applied in a variety of applications such as in computer vision, speech recognition, autonomous driving, medicine, economics and manufacturing.

The term *Machine Learning* was coined in 1959 by Arthur Samuel [29], an IBM employee and pioneer in the field of computer gaming and artificial intelligence. But, the very first idea of combining the way human brain works and machines was published by Warren S. McCulloch, a neuroscientist, and Walter Pitts, a logician in 1943. In their paper, ”A logical calculus of the ideas immanent in nervous activity” [15], they tried to understand how the brain could produce highly complex patterns by using many basic cells that are connected together. So, they gave a highly simplified model of a neuron. After these two contributions, many others followed in order for machine learning and artificial intelligence to become so integrated into our daily lives.

The models build through machine learning algorithms are based on sample data, known as training data, in order to make predictions or decisions without being expressly programmed to do so. This is one among the factors that has contributed to the rapid growth of ML in recent years. There is presently a surplus of data in many fields.

There are three subcategories of machine learning [16]:

- **Supervised** machine learning, which is defined by the use of labeled datasets in the training procedure. For example, an algorithm would be trained with pictures of cats and dogs, all labeled by humans, and the machine would learn ways to identify pictures of cats and digs on its own.
- **Unsupervised** machine learning, which uses unlabeled data. From that data, models discover patterns that help solve clustering or association problems. For example, an unsupervised machine learning program could look through online sales data and identify different types of clients making purchases.
- **Reinforcement** machine learning. In this area, machines learn to achieve a goal in an uncertain and usually complex environment. They are called to take an action, given the state of the environment, and then they take rewards or penalties for the actions they perform. The final goal is to maximize the total reward. Some examples where reinforcement learning is used are game playing and autonomous driving.

Some implementations of machine learning use data and algorithms in a way that they imitate the way of a biological brain works. That area is called Deep Learning and the tools used are named Neural Networks, two terms we will analyse in the next sections.

2.2 Deep Learning

Deep Learning [11] is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. It has the ability to perform really well with unlabeled data. Deep learning can automatically identify the collection of features that separate several types of data from one another and can ingest unstructured data in its raw form (such as text and photos). We can scale machine learning in more interesting ways since it doesn't require human intervention to handle data, unlike machine learning. Progress in fields like computer vision, natural language processing, and speech recognition is mostly attributed to deep learning and neural networks.

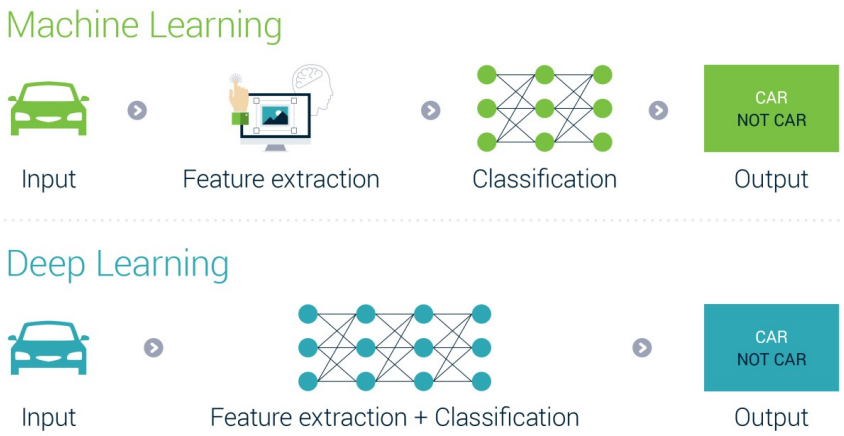

Figure 2.1: Machine Learning vs Deep Learning (Source)

2.3 Neural Networks

Neural networks [15], or artificial neural networks (ANNs), are computing systems inspired by the biological neural networks of humans. Basically, they are consist of several connected units or nodes called artificial neurons. Every node receives signals, then analyzes them and transmits the processed signal to neurons connected to it. Neurons frequently group together into layers. Different layers may modify their inputs in different ways. Signals move through the layers, perhaps more than once, from the first layer to the last layer.

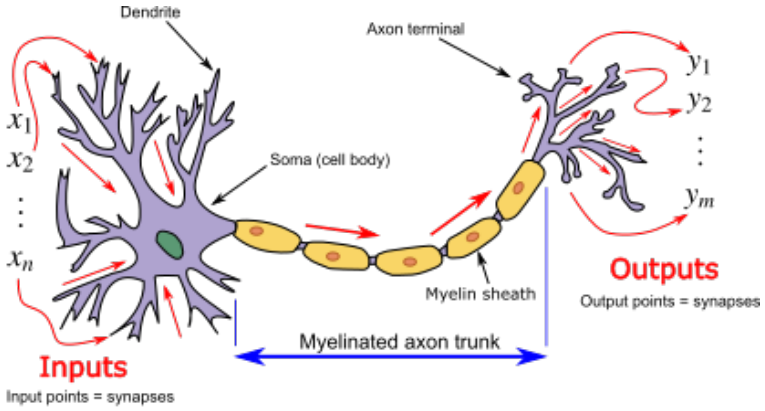

Figure 2.2: Neuron and myelinated axon (Source)

So, ANNs are consist of a node layers, containing an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, connects to another and has an associated weight and threshold. If the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network.

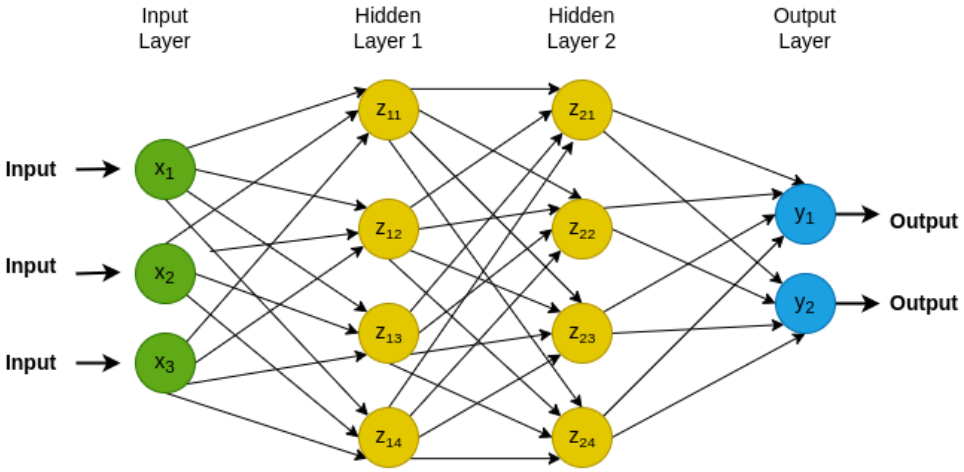

Figure 2.3: Example of Neural Network Architecture

Lastly, the “deep” in deep learning is referred to the depth of layers in a neural network. A neural network that has more than three layers can be considered a deep learning algorithm or a deep neural network. A neural network that only has two or three layers is just a basic neural network.

Another essential component of neural networks is the activation function [31]. An activation function in a neural network defines how a node or nodes in a layer of the network translate the weighted sum of the input into an output. Different activation functions may be used in different regions of the model, and the choice of activation function has a significant impact on the neural network’s capability and performance. Technically, the activation function is used within or after the internal processing of each node in the network, although networks are designed to use the same activation function for all nodes in a layer. All hidden layers typically use the same activation function. Typically, the same activation function is used by all buried levels. The kind of prediction required by the model will determine what activation function is used in the output layer, which is often different from the hidden layers.

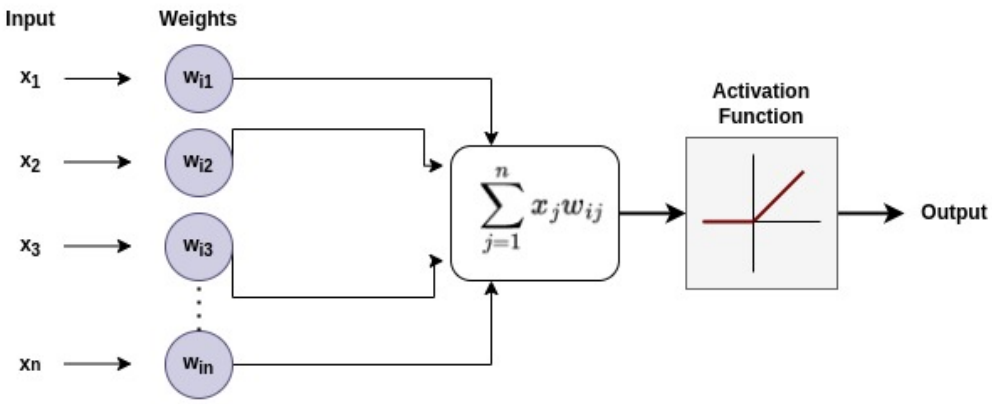

Figure 2.4: Activation Function Usage

2.4 Perceptrons

A perceptron was introduced by F. Rosenblatt [25] in 1958. Is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class[7]. It consists of an input layer, followed by the output layer which is a single neuron. The value of the output neuron can be computed as $f(x) = wx + b$, where $w \in R^{1\times n}$ is the weight vector (signals), $x \in R^{n\times 1}$ is the input vector and $b \in R$ is the bias. The classifier will then choose a class according to the sign of the output neuron. We have:

$$f(x) = \begin{cases} 1 & \text{if wx + b} > 0, \\ 0 & \text{otherwise.} \end{cases}$$

In the context of neural networks, a perceptron is an artificial neuron that uses the Heaviside step function as its activation function. To distinguish it from a multilayer perceptron, a misleading name for a more complex neural network, the perceptron method is also known as the single layer perceptron. The single-layer perceptron is the simplest feedforward neural network and serves as a linear classifier.

For more complicated tasks this model will not perform well, since it only learns a linear decision boundary. In order to overcome this problem, we need to increase the size of our model. So, we add more layers between the input and output layers, followed by a non-linear activation function in order to learn more complex representations of the input data. These new layers are called “hidden” layers and the model with that kind of architecture is called “MultiLayer Perceptron” (MLP) [26].

Specifically, an MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation [28] for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

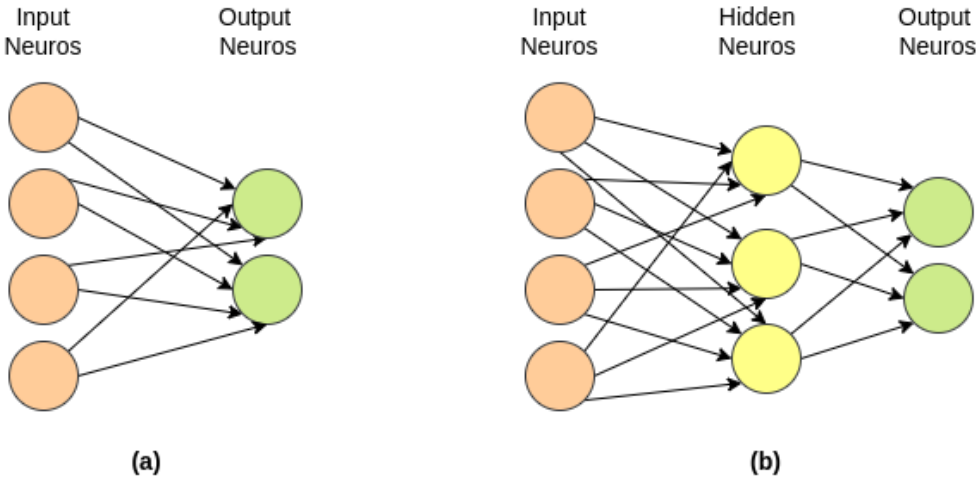

Figure 2.5: Perceptrons. (a) Single layer perceptron; (b) multi-layer perceptron.

2.5 Recurrent Neural Networks

Recurrent neural networks were based on David Rumelhart's work in 1986 [27] and the famous LSTM architecture was invented in 1997 [9] by S. Hochreiter and J. Schmidhuber.

A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data and its connections between nodes form a directed or undirected graph. They have the concept of 'memory' that helps them store the states or information of previous inputs to generate the next output of the sequence. If the network features feedback loops or time delays, the storage can be replaced by another network or graph. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units (GRUs).

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The three gates control the flow of information into and out of the cell, and the cell remembers values across arbitrary time intervals.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem [8] that can be encountered when training traditional RNNs.

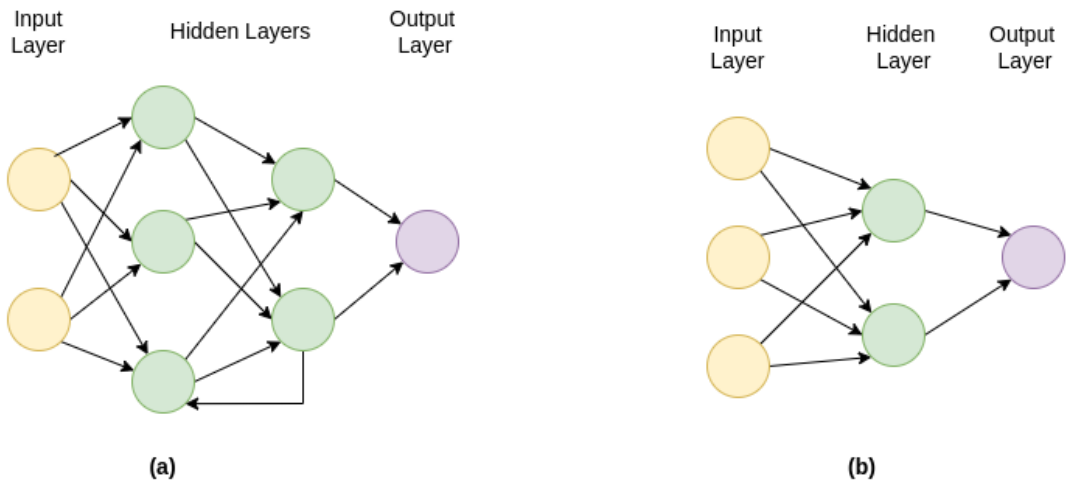

Figure 2.6: (a) Recurrent Neural Networks (b) Non Recurrent Neural Networks

These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (NLP), speech recognition, and image captioning; they are incorporated into popular applications such as Siri, voice search, and Google Translate.

2.6 Transformers

2.6.1 Introduction

Transformers were proposed from Google in the paper “Attention Is All You Need” in 2017 [33]. Now, is the current state-of-the-art technique in the field of NLP.

Specifically, the Transformer Neural Network is a novel architecture that tries to solve sequence-to-sequence problems while handling long-range dependencies. Like Recurrent Neural Networks, transformers are designed to process sequential input data. However, unlike RNNs, they are able to handle the entire input all at once. They make use of an evolving mathematical technique, called attention or self-attention, a mechanism through which the model determine the significance of the input tokens.

The attention mechanism provides context for any position in the input sequence. For instance, if the input data is a natural language sentence, the transformer does not need to process one word at a time. Compared to RNNs, this permits higher parallelization, which reduces the training time and allows training on larger datasets. This led to the development of pretrained systems such as BERT (Bidirectional Encoder Representations from Transformers) [6] and GPT-3 (Generative Pre-trained Transformer 3) [2].

2.6.2 Attention

In neural networks, attention is a technique that imitates cognitive attention. The purpose of this mechanism is to encourage the network to give greater attention to the little but significant portions of the input data by enhancing some and reducing others.

An attention function can be described as mapping a query and a set of key-value pairs of dimension $\sqrt{d_k}$ to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. It can be described by the following equation [33]:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

Q refers to query, K to keys and V stands for values. Additionally, the SoftMax function is applied to the weights in order to have a distribution between 0 and 1. Those weights are then applied to all the words in the sequence that are introduced in V.

2.6.3 Architecture

The input data are parsed into tokens by a byte pair encoding tokenizer, and a word embedding converts each token into a vector. Then, positional information of the token is added to the word embedding.

The original Transformer model uses an encoder–decoder architecture. The encoder consists of encoding layers that process the input iteratively one layer after another, while the decoder consists of decoding layers that do the same thing to the encoder’s output.

Both the encoder and decoder layers have a feed-forward neural network for additional processing of the outputs and contain residual connections and layer normalization steps.

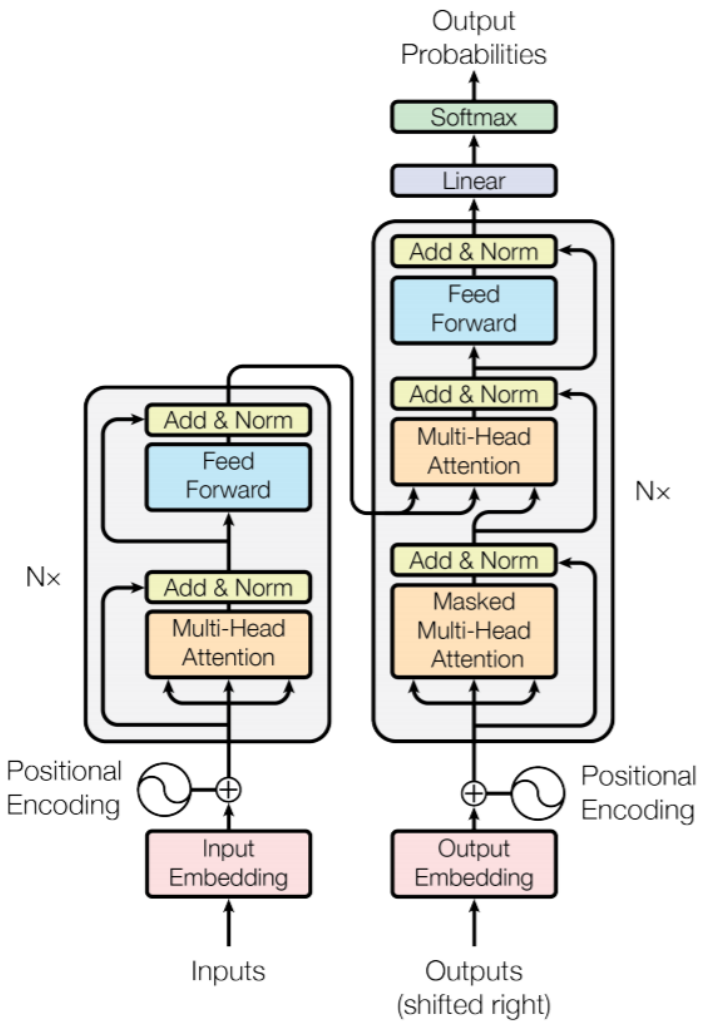

Figure 2.7: Architecture of a transformer [33]

In the above figure, the Encoder is on the left and the Decoder is on the right. Both Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times, which is described by Nx in the figure. We see that the modules consist mainly of Multi-Head Attention and Feed Forward layers.

The function of each encoder layer is to generate encodings that contain information about which parts of the inputs are relevant to each other. It passes its encodings to the next encoder layer as inputs. Each decoder layer does the opposite, taking all the encodings and using their incorporated contextual information to generate an output sequence. To achieve this, each encoder and decoder layer makes use of an attention mechanism.

For each input, attention weighs the relevance of every other input and draws from them to produce the output. Each decoder layer has an additional attention mechanism that draws information from the outputs of previous decoders, before the decoder layer draws information from the encodings.

3. RELATED WORK

3.1 BERT

BERT is a model for natural language processing introduced in Devlin et al [6] in 2018 by Google. It learns bi-directional representations of text and it can significantly improve contextual understanding of unlabeled text across many different tasks. Fine-tuning is achieved only by replacing the final layer of the core model with the one needed for the specific task. This makes BERT suitable for many language understanding tasks like translation, question-answering (Q&A), sentiment analysis, and sentence classification [1].

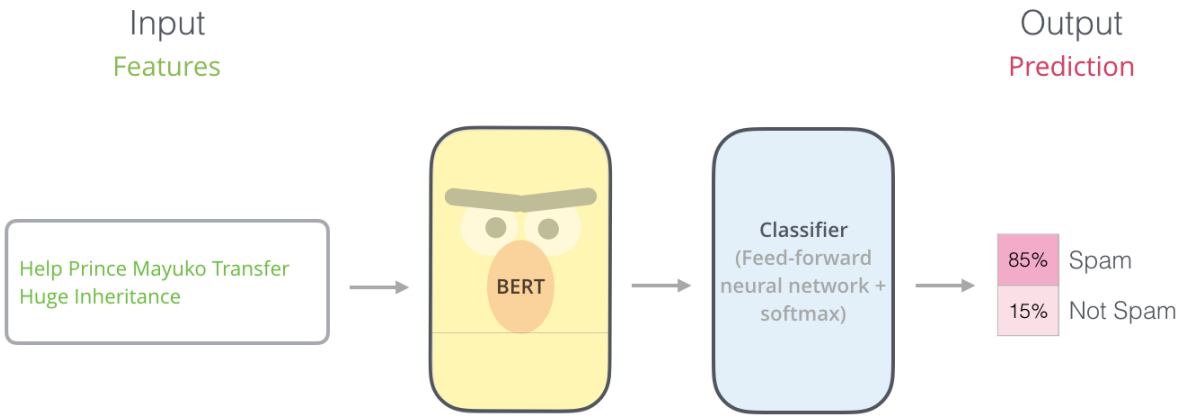

Figure 3.1: NLP task using BERT[1]

3.1.1 How it works

BERT is based on a Transformer architecture. As we mentioned before, Transformers include two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

Directional models read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

The input of the encoder is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.

[1] https://jalammar.github.io/illustrated-bert/

3.1.2 Architecture

As a transformer language model has a variable number of encoder layers and self-attention heads based on the original implementation. The original English-language BERT has two models:

1. $BERT_{BASE}$: 12 encoders with 12 bidirectional self-attention heads, and
2. $BERT_{LARGE}$: 24 encoders with 16 bidirectional self-attention heads

Both models are pre-trained from unlabeled data extracted from the BooksCorpus[35] with 800M words and English Wikipedia with 2,500M words.

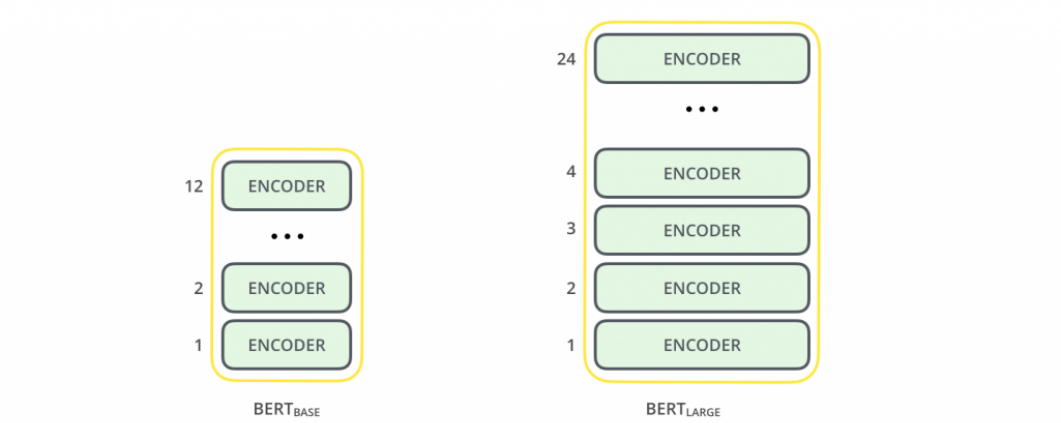

Figure 3.2: BERT architecture overview[1]

3.1.3 Pre-training

BERT's vocabulary uses WordPiece embeddings [34] with a 30,000 token vocabulary. This helps BERT represent every word in existence, by combining learned tokens. As mentioned before, BERT does not use traditional left-to-right or right-to-left language models for pre-training. Instead, it uses the two unsupervised tasks described below.

3.1.3.1 Masked LM

For the pre-training procedure some of the input tokens are randomly masked and then the model attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. Training the language model in BERT is done by predicting 15% of the tokens in the input, that were randomly picked. These tokens are pre-processed as follows:

- 80% are replaced with a "[MASK]" token
- 10% with a random word and
- 10% use the original word.

3.1.3.2 Next Sentence Prediction (NSP)

In some tasks such as Question Answering the model has to understand the relationship between two sentences, an ability difficultly achieved by language modeling. So, BERT was pre-trained in a next sentence prediction task. Specifically, for each training example two sentences A, B are chosen, where 50% of the time B is the actual next sentence that follows A, labeled as $IsNext$ and 50% of the time is a random sentence from the corpus, labeled as $NotNext$.

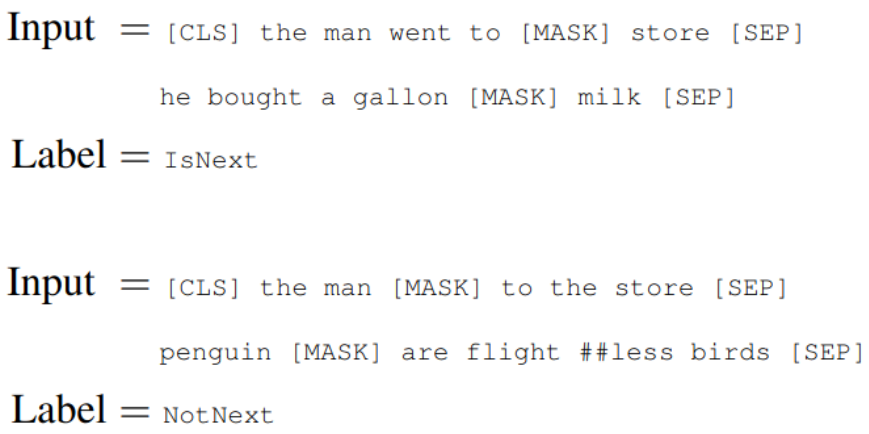

Figure 3.3: Example of Training data (Source)

3.2 GREEK-BERT

GREEK-BERT [10] is a monolingual model for Greek language, introduced by a team in Athens University of Economics and Business, based on the original BERT architecture, trained on 29 GB of Greek text data. The model achieved great performance in Greek PoS tagging, named entity recognition, and natural language inference, outperforming some multilingual Transformer-based language models (M-BERT [1], XLM-R [4]) and shallower established neural methods (BILSTM-CNN-CRF [13], DAM [21]) in the last two tasks.

3.2.1 Data

The pre-training corpora of bert-base-greek-uncased-v1 includes:

- The Greek part of Wikipedia[2], (0.73GB, 0.08 billion tokens)
- The Greek part of European Parliament Proceedings Parallel Corpus[3] (0.38GB, 0.04 billion tokens), and
- The Greek part of OSCAR [19], a cleansed version of Common Crawl[4] (27GB, 2.92 billion tokens).

[2] https://dumps.wikimedia.org/elwiki/
[3] https://www.statmt.org/europarl/
[4] https://commoncrawl.org

3.2.2 Benchmarks

3.2.2.1 Part-of-Speech Tagging

Part-of-Speech (PoS) tagging [22] is a task of labelling each word in a sentence with its appropriate part of speech. Specifically, a PoS tag is a special label assigned to each token (word) in a text corpus to indicate the part of speech and often also other grammatical categories such as tense, number (plural/singular), case etc. PoS tags are used in corpus searches and in text analysis tools and algorithms.

3.2.2.2 Named Entity Recognition

Named-entity recognition (NER) [18] is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.

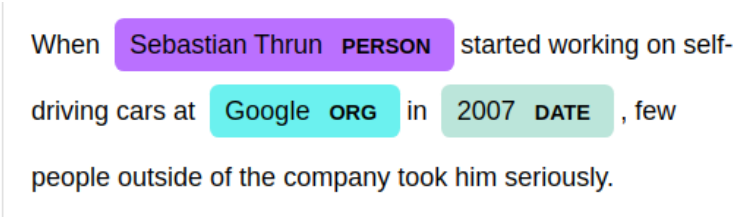

Figure 3.4: NER example (Source)

3.2.2.3 Natural Language Inference

Natural Language Inference (NLI) [5] is a task of determining whether the given "hypothesis" and "premise" logically follow (entailment) or unfollow (contradiction) or are undetermined (neutral) to each other.

$\mathbf{P}^a$	A senior is waiting at the window of a restaurant that serves sandwiches.	Relationship
$\mathbf{H}^b$	A person waits to be served his food.	Entailment
	A man is looking to order a grilled cheese sandwich.	Neutral
	A man is waiting in line for the bus.	Contradiction
^{a}**P**, Premise. ^{b}**H**, Hypothesis.		

Figure 3.5: NLI Example[5]

[5] https://paperswithcode.com/task/natural-language-inference

3.3 RoBERTa

RoBERTa [12] stands for Robustly Optimized BERT Pretraining Approach. It was presented by researchers at Facebook and Washington University in 2019. RoBERTa is based on the architecture of BERT model, but in order to make some improvements on the results, the authors made some simple design changes in its architecture and training procedure. The modifications include:

- Training the model over a longer period of time, using larger batches, and on a longer sequence. BERT is initially trained for 256-sequence batches and 1M steps. The model was trained using 125 steps of 2K sequences and 31K steps of batch size 8K sequences. This has two benefits: the large batches increase end-task accuracy and perplexity on the masked language modelling objective. Additionally, distributed parallel training makes it simpler to parallelize large batches.
- Removing the next sentence prediction objective. The model is trained to predict, using an auxiliary Next Sentence Prediction (NSP) loss, whether the observed document segments originate from the same or other documents. The authors tested numerous versions with and without NSP loss, and they came to the conclusion that removing the NSP loss matches or slightly improves the performance of downstream tasks.
- Dynamically changing the masking pattern applied to the training data. The masking in the BERT architecture is done just once, during the data preprocessing, resulting to a single static mask. To avoid using the single static mask, training data is duplicated and masked 10 times, each time with a different mask strategy over 40 epochs thus having 4 epochs with the same mask. This approach is compared to dynamic masking, which uses a different mask each time input is passed into the model.

3.3.1 Data

The original data of BERT was 16GB, but RoBERTa has 160GB. Specifically, the dataset includes:

- BOOK CORPUS [35] and English Wikipedia dataset: This data also used for training BERT architecture and contains 16GB of text.
- CC-NEWS [6]. This data contains 63 million English news articles crawled between September 2016 and February 2019. The size of this dataset is 76 GB after filtering.
- OPEN WEB TEXT[7]: This dataset contains web content extracted from the URLs shared on Reddit with at least 3 upvotes. The size of it is 38 GB.
- STORIES [32]: This dataset contains a subset of Common Crawl data filtered to match the story-like style of Winograd NLP task. It contains 31 GB of text.

[6] http:web.archive.org/save/http:/commoncrawl.org/2016/10/newsdataset-available
[7] http://web.archive.org/save/http://Skylion007.github.io/OpenWebTextCorpus

3.3.2 Text Encoding

By combining character- and word-level representations, Byte-Pair Encoding (BPE) is able to manage the huge vocabulary that is very common in natural language datasets. BPE depends on subword units rather than whole words, which are derived by statistical analysis of the training corpus. BERT uses a character level BPE vocabulary size of 30K which is learned after preprocessing with heuristic tokenization rules. The team of RoBERTa, uses another implementation of BPE that uses bytes instead of unicode characters at the base subword units. Here BPE subword vocabulary is increased to 50K units. This type of encoding was chosen since it is a universal encoding scheme that doesn't have tokenization or preprocessing requirements.

3.4 Other RoBERTa models

3.4.1 CamemBERT

CamemBERT [14] is a state-of-the-art language model for French based on the RoBERTa architecture pretrained on the French subcorpus of the newly available multilingual corpus OSCAR. CamemBERT was evaluated in four different downstream tasks for French: part-of-speech (POS) tagging, dependency parsing, named entity recognition (NER) and natural language inference (NLI); improving the state of the art for most tasks over previous monolingual and multilingual approaches, which confirms the effectiveness of large pretrained language models for French.

3.4.2 GottBERT

GottBERT [30] is the German version of RoBERTa. The German portion of the OSCAR data set was used as text corpus. For the evaluation, its performance was compared on two Named Entity Recognition (NER) tasks as well as on text classification tasks with existing German single language BERT models and two multilingual ones. GottBERT was pre-trained related to the original RoBERTa model using fairseq. All downstream tasks were trained using hyperparameter presets taken from the benchmark of German BERT. GottBERT surpassed all other tested German and multilingual models in all NER and one text classification tests.

3.4.3 RobBERT

RobBERT [5] is a Dutch language model. Its performance was measured on various tasks as well as its importance of the fine-tuning dataset size. The importance of language-specific tokenizers and the model's fairness were also evaluated . RobBERT improves state-of-the-art results for various tasks, and especially significantly outperforms other models when dealing with smaller datasets.

4. TWO ROBERTA LANGUAGE MODELS FOR GREEK

In this chapter we will represent two models based on the RoBERTa architecture, each with a different size of training data. We will thoroughly examine the training process, the datasets, and in the end we will be able to judge, based on the results, how big role more data play for a model to be effective. We will compare these models with the GREEK-BERT model, mentioned in section 2.2.

4.1 Semi-GrRoBERTa

Semi-GrRoBERTa is a model based on the original architecture of RoBERTa trained on the exact same data as the GREEK-BERT model (29GB), described in detail in section 2.2.1 . The purpose of this experiment is to check if by applying all the changes described in the RoBERTa paper, except adding extra data, the results would be better. The model was trained for 1367 epochs with 13 minutes training time per epoch. So the training process for this model lasted about 5 days. The loss was 9.815 when the model started the training and 0.744 when it ended.

4.2 GrRoBERTa

GrRoBERTa is is also based on RoBERTa and is additionally trained on a much larger volume of Greek data. We wanted to see if this model could be a robustly optimized version of GREEK-BERT. During the pre-training process we applied all the modifications mentioned on the RoBERTa approach.

The model was trained again on the same data as GREEK-BERT, but with the addition of the Greek part of C4 dataset [24] (120GB). The only difference appeared in that extra dataset, is that we needed to do some more pre-proccessing, because it included many emojis, a characteristic that did not existed in the previous datasets. So our final dataset was a size of 149GB, five times bigger than the one of Semi-GrRoBERTa and GREEK-BERT.

GrRoBERTa was trained for 29 epochs with 15 hours and 30 minutes training time per epoch. This gives us approximately 18 days of training. The loss was 15.828 when the model started the training procedure and 1.552 when it ended.

4.3 Server Resources

For the needs of this thesis we used the server facilities of the Cyprus Institute[1]. Both models were trained on 4 NVidia V100 GPUs of 32 GB. The only problem we faced with this facility was the fact that every job we submitted lasted for 24 hours. So, after the end of it we had to commit resources again, which were often not available. Therefore, the actual time of the training procedure, was much longer than the one we mentioned.

[1] `https://hpcf.cyi.ac.cy/index.html`

4.4 Pre-training Procedure

4.4.1 Pre-Process

Firstly, we preprocessed the data in the same way it was handled during the GREEK-BERT preprocessing procedure. Specifically, the data had only lowercase letters, while special symbols and emojis were removed. We later discovered that this step was unnecessary because the tokenizer we used during training already knew how to partition text in order to maximize the coverage of the dictionary size that we chose at the beginning. In fact, this is the power of dynamic tokenizers, that they are based on your data rather than static rules. So, we could have skipped this step, except that the emoji removal is needed in any case.

4.4.2 Encoding

For the pretraining process of both models we based our approach on the instructions[2] of fairseq [20], which is a sequence modeling toolkit that enables researchers and developers to train custom models for translation, summarization, language modeling and other text generation tasks. In this guide there are two basic steps before training the model:

1. Encode data with the GPT-2 BPE
2. Reprocess/binarize the data using the GPT-2 fairseq dictionary

However, in our case we couldn't use the GPT-2 BPE encoder from fairseq, because it is only for English. In order to surpass this obstacle, we talked with the team of UmBERTo [3], a Language model based on RoBERTa architecture for Italian, and they adviced [4] us to use the SentencePiece[5] Tokenizer that is also the same used in Camembert[14]. We first trained the tokenizer to the whole dataset, a procedure that takes several minutes, and then we encoded it. For this process, we used the tools for command line. Specifically:

1. We first trained the tokenizer to all the files of the dataset.

```
# Train SentencePiece Tokenizer on large corpus
spm_train \
    --input=[text files] \
    --max_sentence_length= [ max length of a sentence you accept ]\
    --model_prefix=roberta \
    --vocab_size=[50000 in our case] \
    --model_type=bpe \
    --shuffle_input_sentence=true \
    --pad_id=-1 \
    --input_sentence_size=[ choose a smaller amount of data randomly ]
```

[2] https://github.com/facebookresearch/fairseq/blob/main/examples/roberta/README.pretraining.md

[3] https://github.com/musixmatchresearch/umberto

[4] https://github.com/musixmatchresearch/umberto/issues/2#issuecomment-585894712

[5] https://github.com/google/sentencepiece

2. Then we encoded the data in the format that Fairseq training requires.

```
# Encode Data with SentencePiece Tokenizer
spm_encode \
    --model=roberta.model \ [ model that is from output of sp training ]
    --extra_options=bos:eos \ [ saying that you want begin of sequence and
        end of sequence encoded ]
    --output_format=piece \ [ here you are telling that encoded data will
        be as tokens of spm ]
    < file.txt \ [ data in input]
    > file.bpe [ encoded data in output ]
```

The final bpe file has the following format:

<s> _προκειται _βασικα _για _εγγραφο _πληροφορησης _. </s>
<s> _ποτε _δεν _ειχε _, _αλλωστε _, _γερα _θεμελια _. </s>
<s> _σημερα _, _ολα _τα _κρατη _μελη _αποτελουν _μερος _του _σενγκεν _,
_αν _και _η _συμμετοχη _του _ηνωμενου _βασιλειου _και _της _ιρλανδιας _δεν _ειναι _πληρης _. </s>
<s> _το _σημειωνουμε _, _αγαπητε _συναδελ φε _. </s>

Figure 4.1: Example of pbe encoding

3. After that, in the dictionary that is created, we only had to change the seperator from tab to space, because this is the notation expected by fairseq. The final dictionary has the above format. The *#fiarseq:overwrite* is a message for fairseq, that these particular units might appear again in the file.

<unk> 0 #fairseq:overwrite
<s> 0 #fairseq:overwrite
</s> 0 #fairseq:overwrite
<n> 0 #fairseq:overwrite
_τ -0
ου -1
_α -2
_ε -3
_π -4
ια -5
_σ -6
_κ -7
αι -8
_μ -9
_. -10
ει -11
ικ -12
_, -13
να -14
τα -15

Figure 4.2: Example of the dictionary

4. Finally, after we had all of our data encoded to Byte-Pair Encoding (BPE) format, we preprocessed the encoded data according to fairseq. After this step inside the DATA_DIR the binarized files have been created for every data file.

```
# Preprocess Data
fairseq-preprocess \
    --only-source \
    --srcdict roberta.vocab \
    --trainpref train.bpe \
    --validpref valid.bpe \
    --testpref test.bpe \
    --destdir $DATA_DIR \
    --workers $N_WORKERS
```

4.5 Training Procedure

4.5.1 Training Parameters

Due to the long time and high computing power required by the training, we could not train for 500K steps as the original RoBERTa model. So, we had to choose a smaller number of steps. UmBERTo's [6] team had quite good results with 125K and that's why we chose to adjust our options in line with theirs. Moreover, RoBERTa model had similar performance at both 100K and 500K steps.

```
TOTAL_UPDATES=125000    # Total number of training steps
WARMUP_UPDATES=10000    # Warmup the learning rate over this many updates
PEAK_LR=0.0005          # Peak learning rate, adjust as needed
TOKENS_PER_SAMPLE=512   # Max sequence length
MAX_POSITIONS=512       # Num positional embeddings
MAX_SENTENCES=16        # Number of sequences per batch (batch size)
UPDATE_FREQ=32          # Increase the batch size 32x
ATTENTION_DROPOUT=0.1   # The dropout probability for attention
DROPOUT=0.1             # The dropout probability of the model
WEIGHT_DECAY=0.01       # L2 regularization strength.
OPTIMIZER=ADAM          # Kind of optimizer
```

Also, the final training batch size is 2048 (4GPU * 16 * 32).

[6] https://github.com/musixmatchresearch/umberto

4.5.2 Training Command

After the pre-processing is done and the data files are in the appropriate format, we used the *fairseq-train* command to train the models.

```
fairseq-train --fp16 $DATA_DIR \
--task masked_lm --criterion masked_lm \
--arch roberta_base --sample-break-mode complete --tokens-per-sample
    $TOKENS_PER_SAMPLE \
--optimizer adam --adam-betas '(0.9,0.98)' \
--lr-scheduler polynomial_decay --lr $PEAK_LR --warmup-updates
    $WARMUP_UPDATES --total-num-update $TOTAL_UPDATES \
--dropout $DROPOUT --attention-dropout $ATTENTION_DROPOUT --weight-decay
    $WEIGHT_DECAY \
--batch-size $MAX_SENTENCES --update-freq $UPDATE_FREQ \
--max-update $TOTAL_UPDATES --log-format simple --log-interval 1 \
--save-dir=$SAVE_DIR --skip-invalid-size-inputs-valid-test
```

5. EVALUATION TASKS

For the evaluation of the models we used the tasks of Part-of-Speech (PoS) Tagging and Named Entity Recognition(NER), with the same data as the GREEK-BERT team had. We then compared the performance of our two models with theirs in these two tasks.

5.1 Part-of-Speech (PoS) Tagging

For this task, the Greek Universal Dependencies Treebank (GUDT) [23] [1] was used.The dataset contains 2,521 sentences split in train (1,622), development (403), and test (456) sets. The sentences have been annotated with PoS tags from a collection of 17 universal PoS tags (UPoS)[2]. We ignore the syntactic dependencies of the dataset, since we consider only PoS tagging.

The tags that appear in the dataset are:

Table 5.1: Part-of-Speech Tags

TAG	Meaning
ADJ	adjective
ADP	adposition
ADV	adverb
AUX	auxiliary
CCONJ	coordinating conjunction
DET	determiner
NOUN	noun
NUM	numeral
PART	particle
PRON	pronoun
PROPN	proper noun
PUNCT	punctuation
SCONJ	subordinating conjunction
SYM	symbol
VERB	verb
X	other
_	adposition with determiner, for example ”στην”.

[1] https://github.com/UniversalDependencies/UD_Greek-GDT
[2] https://universaldependencies.org/u/pos/

5.1.1 Data

Example of a the conllu file:

```
# sent_id = gdt-20020206-ep-sessions_031-4
# text = Έτσι, ένας ολλανδός γιατρός ο οποίος πραγματοποίησε έκτρωση σε μία Ιρλανδή δεν χρειάζεται να παραδοθεί στην Ιρλανδία.
1       Έτσι    έτσι    ADV     ADV     _       14      advmod  _       SpaceAfter=No
2       ,       ,       PUNCT   PUNCT   _       14      punct   _       _
3       ένας    ένας    DET     DET     Case=Nom|Definite=Ind|Gender=Masc|Number=Sing|PronType=Art    5       det     _       _
4       ολλανδός        ολλανδός        NOUN    NOUN    Case=Nom|Gender=Masc|Number=Sing       5       nmod    _       _
5       γιατρός         γιατρός         NOUN    NOUN    Case=Nom|Gender=Masc|Number=Sing       16      nsubj:pass      _       _
6       ο       ο       DET     DET     Case=Nom|Definite=Def|Gender=Masc|Number=Sing|PronType=Art    7       det     _       _
7       οποίος  οποίος  PRON    PRON    Case=Nom|Gender=Masc|Number=Sing|Person=3|PronType=Rel       8       nsubj   _       _
8       πραγματοποίησε  πραγματοποιώ    VERB    VERB    Aspect=Perf|Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Act  5       acl:relcl
9       έκτρωση         έκτρωση         NOUN    NOUN    Case=Acc|Gender=Fem|Number=Sing        8       obj     _       _
10      σε      σε      ADP     ADP     _       12      case    _       _
11      μία     ένας    DET     DET     Case=Acc|Definite=Ind|Gender=Fem|Number=Sing|PronType=Art    12      det     _       _
12      Ιρλανδή         Ιρλανδή         PROPN   PROPN   Case=Acc|Gender=Fem|Number=Sing        9       nmod    _       _
13      δεν     δεν     PART    PART    _       14      advmod  _       _
14      χρειάζεται      χρειάζομαι      VERB    VERB    Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin|Voice=Pass  0       root    _
15      να      να      AUX     AUX     _       16      aux     _       _
16      παραδοθεί       παραδίνω        VERB    VERB    Aspect=Perf|Mood=Ind|Number=Sing|Person=3|VerbForm=Fin|Voice=Pass     14      csubj   _       _
17-18   στην    _       _       _       _       _       _       _       _
17      σ       σε      ADP     AsPpSp  _       19      case    _       _
18      την     ο       DET     AtDf    Case=Acc|Gender=Fem|Number=Sing       19      det     _       _
19      Ιρλανδία        Ιρλανδία        PROPN   PROPN   Case=Acc|Gender=Fem|Number=Sing        16      obl     _       SpaceAfter=No
20      .       .       PUNCT   PUNCT   _       14      punct   _       _
```

Figure 5.1: Example of the Conllu File from PoS Tagging Dataset

For this downstream task we used only the first and fourth columns that refer to each word of the sentence and its tag respectively. We copied each word and each tag to text files with a space between them. The end of each sentence is marked with an empty line. Here is an example of a final file:

```
αυτό PRON
είναι AUX
σωστό ADJ
. PUNCT

έτσι ADV
, PUNCT
ένας DET
ολλανδός NOUN
γιατρός NOUN
ο DET
```

Figure 5.2: Part of the test.txt of PoS Tagging

5.2 Named Entity Recognition (NER)

For the second downstream task we used the two unpublished datasets developed by I. Darras [3]and A. Romanou [4] that were used by the GREEK-BERT team. The first one is based on the conllu format and the second one is a json file. The two datasets contained 1,798 and 2,521 sentence in respect. We then merged them to one with 4,189 unique sentences. As the GREEK-BERT team did, we used only the labels of *Person, Organization and Location*, because these are the only common entity types across the two datasets.

[3]Part of his project for Google Summer of Code 2018 (`https://github.com/eellak/gsoc2018-spacy`)
[4]She annotated documents with named entities for another project (`http://greekner.me/info`)

5.2.1 Data

The format of the final dataset should be with Inside–outside–beginning (IOB) tagging. The IOB format is a common tagging format for tagging tokens in a chunking task in computational linguistics. An O tag indicates that a token belongs to no chunk. The B- prefix before a tag indicates that the tag is the beginning of a chunk that immediately follows an other chunk without O tags between them. It is used only in that case: when a chunk comes after an O tag, the first token of the chunk takes the I- prefix. Another similar format which is widely used is IOB2 format, which is the same as the IOB format except that the B- tag is used in the beginning of every chunk and this is the format used for this project. The labels used are as follows: B-LOC, I-LOC, B-PER, I-PER, B-ORG, I-ORG, O. For example:

Text: "Anna wants to study in Los Angeles"

Anna B-PER
wants O
to O
study O
in O
Los B-LOC
Angeles I-LOC

Figure 5.3: Example of IOB2 format

Dataset from A. Romanou

This dataset has already separated files of train, test and validation sets in conllu format. For these files, we read every line and wrote the word and its label to a text file with a space in between. After the end of each sentence we added an empty line. The labels are in the feature NameType from the 6th column. Here is an example of a data file:

```
# newdoc id = gdt-20090406-apogevmatini-news_piravlos_vorias_koreas
# sent_id = gdt-20090406-apogevmatini-news_piravlos_vorias_koreas-1
# text = «Παγκόσμια απειλή» η εκτόξευση πυραύλου από τη Βόρεια Κορέα
1    «    «    PUNCT    PUNCT    _    3    punct    _    SpaceAfter=No
2    Παγκόσμια    παγκόσμιος    ADJ    ADJ    Case=Nom|Gender=Fem|Number=Sing    3    amod    _    _
3    απειλή    απειλή    NOUN    NOUN    Case=Nom|Gender=Fem|Number=Sing    0    root    _    SpaceAfter=No
4    »    »    PUNCT    PUNCT    _    3    punct    _    _
5    η    ο    DET    DET    Case=Nom|Definite=Def|Gender=Fem|Number=Sing|PronType=Art    6    det    _    _
6    εκτόξευση    εκτόξευση    NOUN    NOUN    Case=Nom|Gender=Fem|Number=Sing    3    nsubj    _    _
7    πυραύλου    πύραυλος    NOUN    NOUN    Case=Gen|Gender=Masc|Number=Sing    6    nmod    _    _
8    από    από    ADP    ADP    _    11    case    _    _
9    τη    ο    DET    DET    Case=Acc|Definite=Def|Gender=Fem|Number=Sing|PronType=Art    11    det    _    _
10    Βόρεια    βόρειος    ADJ    ADJ    Case=Acc|Gender=Fem|Number=Sing|NameType=GPE    11    amod    _    _
11    Κορέα    Κορέα    PROPN    PROPN    Case=Acc|Gender=Fem|Number=Sing|NameType=GPE    6    nmod    _    _
```

Figure 5.4: Example of the Conllu File

Dataset from I. Darras

This dataset includes one json file for training data with the following format:


```
{'spans': [{'start': 3,
   'token_start': 1,
   'token_end': 1,
   'label': 'GPE',
   'end': 6}],
 '_input_hash': -149672055,
 '_task_hash': -454002367,
 'text': 'Οι ΗΠΑ ετοιμάζονται να «ανοίξουν πυρ» εναντίον όλου του κόσμου με τους δασμούς αυτούς, είπε ο εκπρόσωπος.',
 'answer': 'accept',
 'tokens': [{'start': 0, 'text': 'Οι', 'id': 0, 'end': 2},
  {'start': 3, 'text': 'ΗΠΑ', 'id': 1, 'end': 6},
  {'start': 7, 'text': 'ετοιμάζονται', 'id': 2, 'end': 19},
  {'start': 20, 'text': 'να', 'id': 3, 'end': 22},
  {'start': 23, 'text': '«', 'id': 4, 'end': 24},
  {'start': 24, 'text': 'ανοίξουν', 'id': 5, 'end': 32},
  {'start': 33, 'text': 'πυρ', 'id': 6, 'end': 36},
  {'start': 36, 'text': '»', 'id': 7, 'end': 37},
  {'start': 38, 'text': 'εναντίον', 'id': 8, 'end': 46},
  {'start': 47, 'text': 'όλου', 'id': 9, 'end': 51},
  {'start': 52, 'text': 'του', 'id': 10, 'end': 55},
  {'start': 56, 'text': 'κόσμου', 'id': 11, 'end': 62},
  {'start': 63, 'text': 'με', 'id': 12, 'end': 65},
  {'start': 66, 'text': 'τους', 'id': 13, 'end': 70},
  {'start': 71, 'text': 'δασμούς', 'id': 14, 'end': 78},
  {'start': 79, 'text': 'αυτούς', 'id': 15, 'end': 85},
  {'start': 85, 'text': ',', 'id': 16, 'end': 86},
  {'start': 87, 'text': 'είπε', 'id': 17, 'end': 91},
  {'start': 92, 'text': 'ο', 'id': 18, 'end': 93},
  {'start': 94, 'text': 'εκπρόσωπος', 'id': 19, 'end': 104},
  {'start': 104, 'text': '.', 'id': 20, 'end': 105}]}
```


Figure 5.5: Example of the Json File

As we see, the spans section is needed, because it includes the labels of some tokens. We can find the word each label refers to, by the 'start' and the 'end' of each token. In the above example we have only the label for a word that has at start the number 3 and at the end the number 1. If we check in the section of tokens, we can see that it refers to the word 'ΗΠΑ'. So, we conclude that 'ΗΠΑ' is labeled as GPE, and because of the BIO2 format to B-LOC. All the other words will be labeled as O - other.

Final Dataset

The first dataset was already splitted to training, validation and test sets. According to the GREEK-BERT paper, we combined the training part of A. Romanou's dataset with the entire dataset from I. Darras in order to create the new training set. For validation and test sets, we used the validation and test files of the first set respectively. The final dataset includes three text files: train.txt (1,1 MB), dev.txt (135 KB), test.txt (135 KB) . These files have the word in each line on the left, then a space and then the corresponding label. Each sentence stands out from the other, with a blank line between them.

6. EXPERIMENTAL RESULTS

In this section, we present the scores of the two models and we compare their performance to the GREEK-BERT model. Among the models we created, we expect GrRoBERTa to perform better since the size of the data plays a decisive role in such cases.

6.1 GREEK-BERT results

In the paper of AUEB team the micro-F1 scores were 98.1 and 85.7 respectively for the two tasks. We also ran in our own the GREEK-BERT model in the two downdtream tasks and reported very similar results to the paper:

Table 6.1: GREEK-BERT Micro-F1

Model	PoS Tag	NER
GREEK-BERT	98.2	86.6

For this procedure we used the same fine-tuning functions as they did in the repository [1] they published. We used the best parameters that occurred from the tuning function, then trained the BERT model with these and the best scores we had are shown above.

6.2 PoS Tagging

Table 6.2 reports the PoS tagging results. All models have comparable performance, but GREEK-BERT is marginally (+0.6%) better than GrRoBERTa and 2% better than Semi-GrRoBERTa. The fact that all models achieve high scores can be explained by the observation that the correct PoS tag of a word in Greek can often be determined by considering mostly the word's suffix, or for short function words (e.g., determiners, prepositions) the word itself and most rarely the word's context.

Table 6.2: Models Micro-F1 in PoS Tagging

Model	PoS Tag
GREEK-BERT	**98.2**
Semi-GrRoBERTa	96.2
GrRoBERTa	97.6

We certainly can not characterize this task difficult, because all three model have very close scores and all above 95%. For a more complete comparison we also report results in Table 6.3 per PoS tag for the two best models, i.e., GREEK-BERT and GrRoBERTa. We notice that again the performance is pretty close between the models.

[1] `https://github.com/nlpaueb/greek-bert`

Table 6.3: F1 scores per PoS tag for the two best models (GREEK-BERT, GrRoBERTa)

Part-of-Speech Tag	GREEK-BERT	GrRoBERTa
ADJ	95.6 ± 0.26	94.8
ADP	99.7 ± 0.07	97.9
ADV	97.2 ± 0.34	89.1
AUX	99.9 ± 0.15	97.8
CCONJ	99.6 ± 0.24	97.4
DET	99.8 ± 0.08	97.4
NOUN	97.9 ± 0.28	97.9
NUM	92.7 ± 1.14	91.4
PART	100.0 ± 0.00	97.2
PRON	98.8 ± 0.25	96.6
PROPN	86.0 ± 1.03	89.7
PUNCT	100.0 ± 0.00	99.9
SCONJ	99.4 ± 0.56	97.5
VERB	99.3 ± 0.13	98.6
X	77.3 ± 1.32	79.4

6.3 NER

The NER results are shown in Table 6.4. We find that GrRoBERTa outperforms the other methods. It outperforms Semi-GrRoBERTa by 9.9% and GREEK-BERT by about 1.5%. The NER task is clearly more difficult than PoS tagging, as evidenced by the near-perfect performance of all PoS tagging methods (Table 6.3), compared to the far from perfect performance of all NER methods (Table 6.4). Because the NER task is more difficult, better methods have more opportunity to distinguish themselves from weaker methods, and GrRoBERTa clearly outperforms the other methods, with GREEK-BERT coming in second. In Table 6.5, we evaluate the two best NER models based on entity type. When

Table 6.4: Models Micro-F1 in NER

Model	NER
GREEK-BERT	86.6
Semi-GrRoBERTa	78.2
GrRoBERTa	**88.1**

it comes to predicting people both models outperform. GREEK-BERT performs better on people and locations, while GrRoBERTa performs a lot of better on organization. GREEK-BERT struggles in this case, because some organizations often contain names of people or locations that appear in italics. One problem, GrRoBERTa seems to overcome.

Table 6.5: F1 scores per Entity Type for the two best models (GREEK-BERT, GrRoBERTa)

Entity Type	GREEK-BERT	GrRoBERTa
PERSON	88.8 ± 3.06	87.5
ORGANIZATION	69.6 ± 4.28	86
LOCATION	88.4 ± 0.88	83.5

6.4 Comments

According to the results we presented in the above sections the two best models are, as we expected, GrRoBERTa and GREEK-BERT. The RoBERTa-based model performs better in the second task and the BERT-based model at the first one. At first glance, one might call the two models equivalent. However, as mentioned above, the PoS tagging task is an easy problem for Greek Language models and the Named Entity Recognition task cannot be considered to have the same level of difficulty. Consequently, since GrRoBERTa can perform better in more difficult problems, we can conclude that it is a more powerful model for Greek Language.

Unfortunately, in the world of Artificial Intelligence and Machine Learning in particular, we are not only concerned with the final performance of the model, but the training time and the amount of data required play a very important role too. As mentioned in the previous chapters the time required to train the model based on RoBERTa's architecture is quite long, given the amount of data we had. For the training of the GREEK-BERT model only 29 GB were needed, while for ours 149 GB and the final scores did not have a huge difference. However, the pre-training process is done once. So, to get a better model it might be worth spending a little more time.

7. CONCLUSION

In this thesis we developed two language models for handling Greek and compared their performance to an existed model with similar architecture.

In the beginning of this process, we examined the datasets that GREEK-BERT was evaluated, then we processed them so that we could ran the model in ourselfs and check its performance. After that, we collected and cleaned all the data needed for our models. Then, after conducting study on our options for training the models, we chose the fairseq strategy. The next stage was to choose a server where we could train our models, and then, after going through the necessary steps, process and encode our data. At this point we were ready to start the training process. After that phase was done, we developed the functions needed in order to evaluate our model with the same datasets of GREEK-BERT, so that we could compare all models to the same tasks.

We came to the conclusion that GrRoBERTa is a more potent model for Greek language since it could perform better in more challenging problems. However, in the field of artificial intelligence and machine learning in particular, we are not only concerned about the model's final performance; training time and the amount of data needed also play a significant role.

Finally, we may have succeeded in creating a model that performs better than an existing one, but as researchers we must ask ourselves what we have sacrificed on the altar of performance and whether this sacrifice is ultimately worth it. In any case, we ourselves gained a lot from this process and we hope that this small stone in the branch of science will help for even greater achievements.

ABBREVIATIONS - ACRONYMS

ML	Machine learning
AI	Artificial Intelligence
NLP	Natural Language Processing
ANN	Artificial Neural Network
MLP	MultiLayer Perceptron
LSTM	Long Short-term Memory
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
BERT	Bidirectional Encoder Representations from Transformers
GPT-3	Generative Pre-trained Transformer 3
Q&A	Question & Answering
PoS	Part of Speech tagging
NER	Named Entity Recognition
NLI	Natural Language Inference
RoBERTa	Robustly Optimized BERT Pretraining Approach
NSP	Next Sentence Prediction
BPE	Byte-Pair Encoding
IOB	Inside–Outside–Beginning

BIBLIOGRAPHY

[1] Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610, nov 2019.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Gobinda G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003.

[4] Alexis CONNEAU and Guillaume Lample. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] Pieter Delobelle, Thomas Winters, and Bettina Berendt. Robbert: a dutch roberta-based language model, 2020.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[7] Y. Freund and R. E Schapire. Large margin classification using the perceptron algorithm. In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT' 98). ACM Press., 1998.

[8] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[10] John Koutsikakis, Ilias Chalkidis, Prodromos Malakasiotis, and Ion Androutsopoulos. GREEK-BERT: The greeks visiting sesame street. In *11th Hellenic Conference on Artificial Intelligence*. ACM, sep 2020.

[11] Bengio-Y. Hinton G. LeCun, Y. Deep learning. *Nature 521*, page 436–444, 2015.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[13] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics.

[14] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suá rez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a tasty french language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[15] Pitts W. McCulloch, W.S. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133, 1943.

[16] Afshin Rostamizadeh Mehryar Mohri and Ameet Talwalkar. Foundations of machine learning. MIT Press, Second Edition, 2018.

[17] Tom Mitchell. Machine learning. New York: McGraw Hill. ISBN 0-07-042807-7. OCLC 36417892., 1997.

[18] Behrang Mohit. *Named Entity Recognition*, pages 221–245. 03 2014.

[19] Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In Piotr Bański, Adrien Barbaresi, Hanno Biber, Evelyn Breiteneder, Simon Clematide, Marc Kupietz, Harald Lüngen, and Caroline Iliadi, editors, *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom, July 2019. Leibniz-Institut für Deutsche Sprache.

[20] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[21] Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas, November 2016. Association for Computational Linguistics.

[22] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset, 2011.

[23] Prokopis Prokopidis and Haris Papageorgiou. Universal dependencies for greek. In *UDW@NoDaLiDa*, 2017.

[24] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[25] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386–408., 1958.

[26] Frank. x. Rosenblatt. Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Spartan Books, Washington DC, 1961.

[27] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[28] Hinton G. Williams R. Rumelhart, D. Learning representations by back-propagating errors. Nature 323, 533–536 ()., 1986.

[29] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[30] Raphael Scheible, Fabian Thomczyk, Patric Tippmann, Victor Jaravine, and Martin Boeker. Gottbert: a pure german language model, 2020.

[31] Anidhya Athaiya Siddharth Sharma, Simone Sharma. Activation functions in neural networks. International Journal of Engineering Applied Sciences and Technology, Vol. 4, Issue 12, ISSN No. 2455-2143, Pages 310-316, 2020.

[32] Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning, 2018.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[34] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016.

[35] Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015.