NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

The large language model GreekLegalRoBERTa

Vasileios E. Saketos

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

ATHENS

February 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Το μεγάλο γλωσσικό μοντέλο GreekLegalRoBERTa

Βασίλειος Ε. Σακέτος

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

ΑΘΗΝΑ

Φεβρουάριος 2023

BSc THESIS

The large language model GreekLegalRoBERTa

Vasileios E. Saketos
S.N.: 1115201800168

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

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Το μεγάλο γλωσσικό μοντέλο GreekLegalRoBERTa

Βασίλειος Ε. Σακέτος

Α.Μ.: 1115201800168

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

ABSTRACT

We develop GreekLegalRoBERTa, a large language model trained on Greek legislation. We show that our model surpasses the performance of both GreekLegalBERT and GreekBERT in two tasks involving Greek legal documents: named entity recognition and multi-class legal topic classification. We view our work as a contribution to the study of domain-specific NLP tasks in low-resource languages, like Greek, using modern NLP techniques and methodologies.

SUBJECT AREA: Artificial Intelligence

KEYWORDS: RoBERTa, BERT, Neural Networks, Natural Language Processing, legal documents

ΠΕΡΙΛΗΨΗ

Σε αυτή την πτυχιακή αναπτύσσουμε το μοντέλο GreekLegalRoBERTa πάνω σε κείμενα Ελληνικής νομοθεσίας. Έπειτα αξιολογούμε την απόδοση του μοντέλου μας σε 2 προκλήσεις νομικού περιεχομένου οι οποίες είναι η αναγνώριση ονοματισμένων οντοτήτων και ο διαχωρισμός νομικών κειμένων σε κατηγορίες. Στο τέλος αποδεικνύουμε ότι η απόδοση του μοντέλου μας ξεπερνάει την απόδοση των μοντέλων GreekLegalBERT και GreekBERT. Η πτυχιακή αυτή αποτελεί μία συνεισφορά στο πεδίο της επεξεργασίας φυσικής γλώσσας συγκεκριμένου περιεχομένου. Η συνεισφορά αυτή είναι ιδιαίτερα σημαντική για μια γλώσσα περιορισμένου περιεχομένου όσον αφορά την επεξεργασίας φυσικής γλώσσας όπως τα ελληνικά.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Θεματική Περιοχή

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: RoBERTa, BERT, Νευρωνικά Δίκτυα, Επεξεργασία Φυσικής Γλώσσας, νομικά έγγραφα

This thesis is dedicated to my family. I am so grateful for your love, support and encouragement. Thank you for being next to me and supporting me in every goal I set.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Manolis Koubarakis for his guidance in my Ai journey first as a professor and then as a supervisor. Thank you for inspiring to explore NLP.

I would also like to thank my research supervisor Despina-Athanasia Pantazi for helping me to overcome every obstacle in this research and for her guidance through constructive suggestions.

CONTENTS

LIST OF FIGURES

PREFACE

Eight months ago I attended an NLP workshop hosted by the Hellenic Artificial Intelligence Society and Manolis Koumparakis. The speaker was Omar Sanseviero. In this workshop, I got inspired by the development opportunities trough the Hugging Face framework. Soon after that, I started to explore different NLP models and learning more about the NLP domain. In this research, I came across the outstanding paper of Meta called RoBERTa. This paper demonstrate that simplicity is the key to better results. As Mr. Koumparakis always highlighted in his lessons "The power is in simplicity". So After reading this paper my goal was to learn every detail of RoBERTa and the best way to do that was to implement it. Then, I realized that producing a new RoBERTa model is an ideal project for my Bachelor's thesis. Furthermore,I also dot inspired by the related work of my Professor team on Legal NLP tasks. Consequently, I decided to utilize my Professor team background and built a RoBERTa model on legal tasks. When I reached out to Mr. Koumparakis he said "Go ahead" and that's what I did!

1. INTRODUCTION

Nowadays NLP [5] applications are becoming part of our life. Text Autocomplete, Machine Translation and now Text To Image conversion are tools we use every day. Within the past few years, a great number of remarkable models exhibiting significant language performance have been introduced. However, a major concern that emerges with these models is that they are mainly trained in English, rendering them unavailable in other languages. Furthermore, despite achieving superhuman performance in various natural language tasks they tend to under perform in specific tasks such as Legal and Biological. Consequently, the responsibility of training and launching a new language model falls upon the users of that language.

In this thesis, we develop a model specifically designed to solve legal tasks. To achieve our goal we are going to use RoBERTa approach [27]. RoBERTa is a sophisticated version of BERT [13]. In the original paper the RoBERTa team discovered that BERT was significantly undertrained and proposed an improved recipe for training BERT. This approach is based on simplicity and shows that a simpler training method can lead to significant growth in performance. To train our model, we utilize a legal dataset obtained from the Nomothesi@ platform [7]. Nomothesi@ is a linked data platform that makes Greek legislation easily accessible to the public, law professionals, and application developers. This thesis is divided into three chapters:

- In chapter 2, we provide the historical background of our research. We provide information, structure, and applications of the postRoBERTa methods.
- In chapter 3, we analyze the pretraining process of RoBERTa and the modifications we made step by step to adapt it to our problem.
- In chapter 4, we present our conclusions and future experiments.

2. BACKGROUND AND RELATED WORK

In this chapter we analyse the related work the background and the models that lead to the development of the RoBERTa.

2.1 Artificial Neural Networks (ANNs)

The goal of an ANN [1] is to solve problems that are hard even for humans to solve. Artificial Neural Networks or ANNs is a construction inspired by human brain. The human brain consists of nodes called neurons connected to each other by connections called synapses or edges. The output of each neuron is computed by a non linear function of the sum of its inputs. Neurons and edges typically have a weight. During training, ANN adjusts these weights to perform better in a specific task. The training follows a procedure called backpropagation.

2.2 Feed Forward ANNs

Feed Forward ANNs [11] is the first and the simplest form of an ANN. In this network, the information moves in only one direction forward from the input nodes, through the hidden nodes (if any), and to the output nodes as shown in figure 2.1. There are no cycles or loops in the network. Perceptron [6] was an example of an FFANN. Perceptron was created by ABM as software to be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Even though it seemed promising in the beginning it was quickly proved that Perceptrons could not be trained to recognize many classes of patterns. This caused the field of neural network research to stagnate for many years before it was recognized that a Feed Forward Neural Network with two or more layers (also called multilayer Perceptron) had greater processing power than Perceptron with one layer. Single layered Perceptrons are capable of learning linear patterns but multilayer Perceptron is even capable of solving non linear problems.

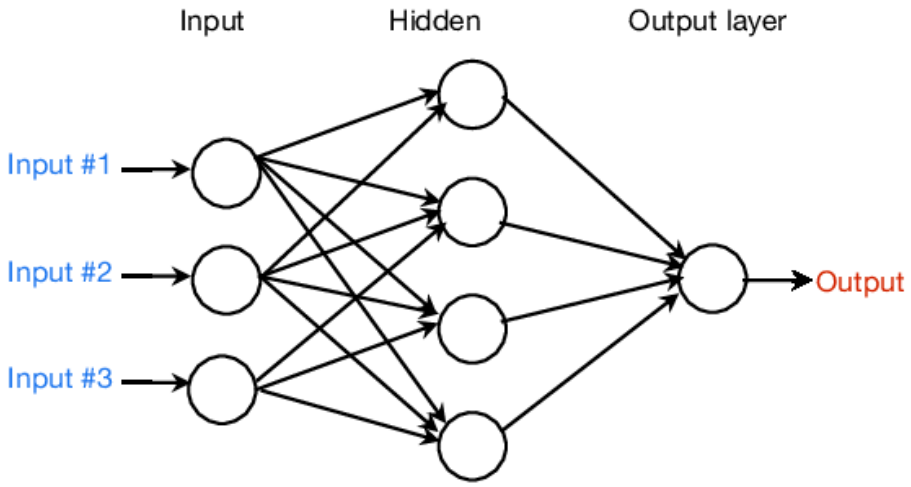

Figure 2.1: Feed Forward Neural Networks

2.3 Recurrent Neural Networks (RNNs)

RNNs [12] is a generalization of traditional FFANN and is among the most promising algorithms in use because it is the only one with internal memory. Because of their internal memory, RNNs can remember important things about the input they received. That allows them to be very precise in predicting future input. This is why they're the preferred for sequential data like time series, speech, text, financial data, audio, video, weather and much more. Recurrent Neural Networks can form a much deeper understanding of a sequence and its context compared to other algorithms. Due to their efficient construction that figure 2.2 depicts. RNNs can handle sequences of variable length while keeping the model structure stable.

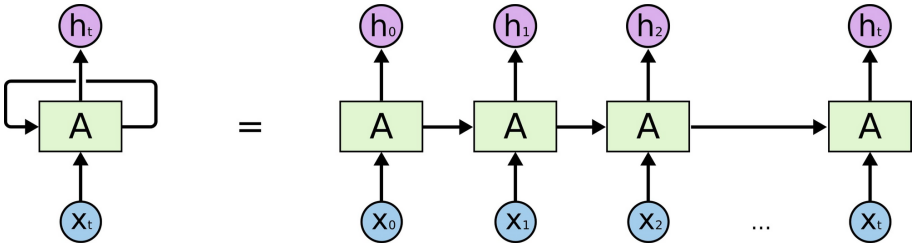

Figure 2.2: Recurrent Neural Networks (RNNs)

Despite the advantages of RNNs 3 major problems appeared :

1. Loops and input-output interaction in RNNs caused problems in the way the chain rule and backpropagation function. Backward Propagation Through Time, a gradient-based technique for training an unrolled RNN, resolved those issues. Basically, a RNN is viewed as a sequence of neural networks trained one after the other with backpropagation.

2. Exploding gradients, is the case that the algorithm assigns very high value to some of the weights. This issue can be easily solved by gradient clipping. Gradient clipping involves forcing the gradient values (element-wise) to a specific minimum or maximum value if the gradient exceeded an expected range.

3. Vanishing gradients, happens when the gradients shrinks causing the model unable to propagate relevant information across distant time steps. In contrast to exploding gradients, vanishing gradients was a harder problem to be solved and lead to the appearance of more powerful models.

2.4 Bidirectional RNNs

RNNs are designed to keep track of past information. But what if the current state information is future dependent? In this case, we add another layer to our model that transmits the information backward as shown in figure 2.3.

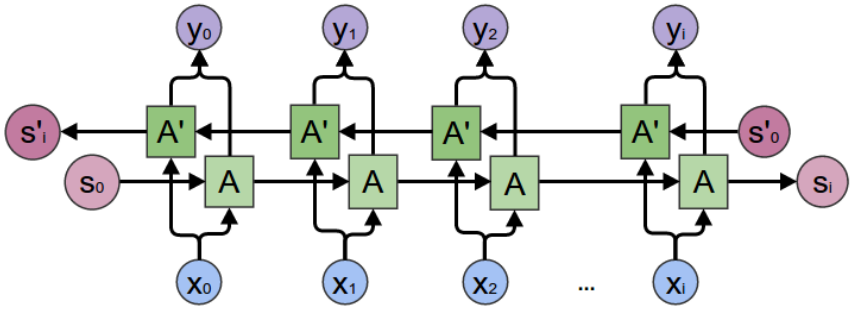

Figure 2.3: Bidirectional RNNs

2.5 Long Short Term Memory (LSTMs)

LSTMs [12] were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs.

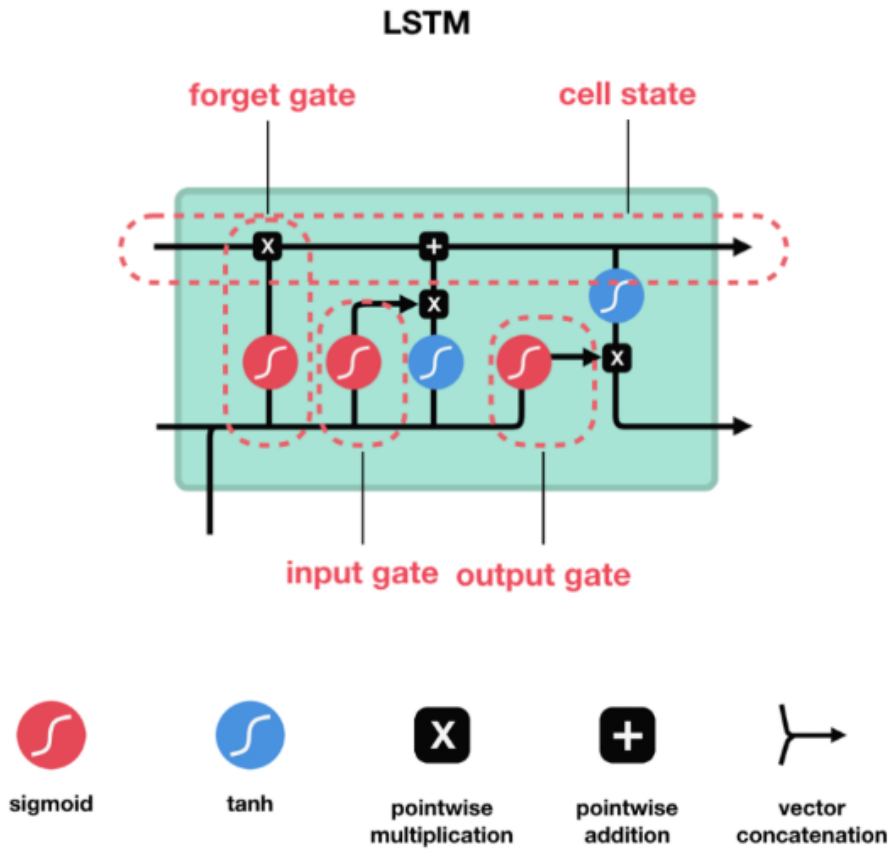

Figure 2.4: Long Short Term Memory (LSTMs)

A common LSTM unit is shown in figure 2.4 and it is composed of an input gate, an output gate and a forget gate.

1. Forget Gate: Decides how much of the past information the model should remember.
2. Input Gate: It is responsible for determining how much of the incoming information should be stored in the memory cell and how much should be discarded.
3. Output Gate: Decides which parts of the current cell are contributing to the output.

In theory, LSTMs can keep track of arbitrary long term dependencies in the input sequences. In practice, this number is limited to 10.

2.6 Attention mechanism

Attention [2] is a technique that enhances some parts of the input data while diminishing other parts. The main idea is that the network should focus more on the small, but important, parts of the data. Learning which part of the data is more important than another depends on the context, and this is trained by gradient descent. The general idea is depicted in figure 2.5. Given a sequence of tokens, a neural network computes a soft weight w_i for each token i with the property that w_i is non negative and $\sum_i w_i = 1$.

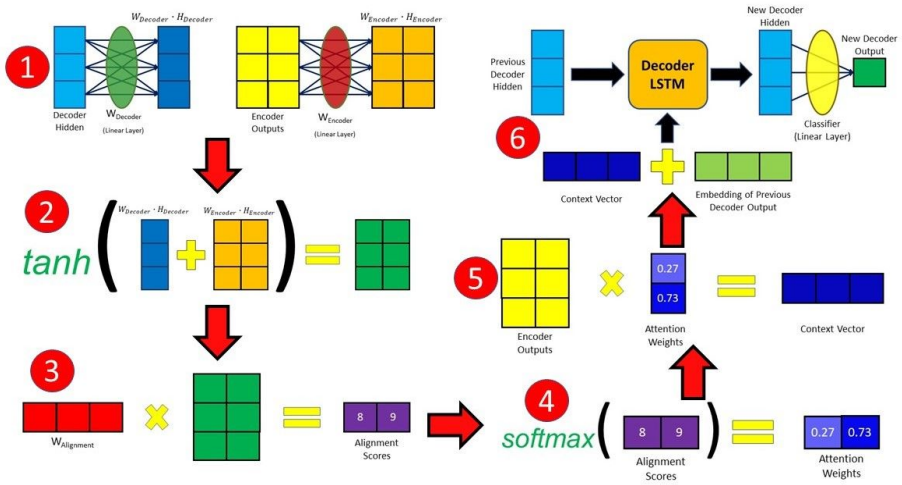

Figure 2.5: Attention mechanism

2.7 Encoder Decoder models

The following architecture is depicted in figure 2.6 was introduced in the paper Attention Is All You Need [24] and inspired the generation of significant models for various machine learning task. A present a small sample of them on the table 2.1. The model consists of two blocks:

- Encoder (left): The Encoder receives an input and builds a representation of it's features. The goal o the model is to acquire an understanding of the input.
- Decoder (right): The Decoder uses the Encoder's representation (features) along with encoder's inputs to generate a target sequence. The goal of the decoder is to generate and output for a given input.

Encoder Decoder models (also called sequence to sequence models) use both parts of the Transformer architecture. This architecture was originally designed for translation but it can be also used for summarization as well as language generation. During training, the Encoder receives inputs (sentences) in a certain language, while the Decoder receives the same sentences in the desired target language. In the Encoder, the attention layers can keep track of all the words in a sentence. This is crucial for the models performance since

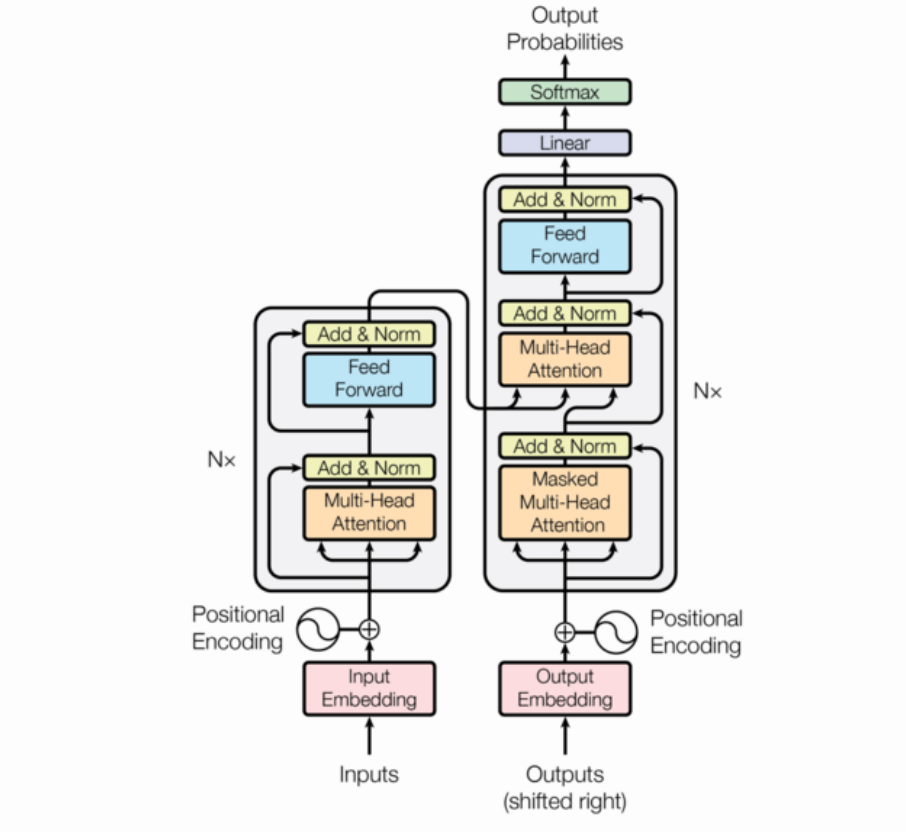

Figure 2.6: Encoder Decoder architecture

a word in a sentence can be both past and future dependent. The Decoder, however, works sequentially and it gets as an input only the already translated words. For example, when we have predicted the first three words of the translated target, we give them to the Decoder which then uses all the inputs of the Encoder to try to predict the fourth word.

2.7.1 Encoder models

Encoder models use only the Encoder of a Transformer model.The pretraining of these models usually involves techniques like masked language modeling and causal language modeling that we will analyze further in this thesis.

2.7.2 Decoder models

Decoder models use only the Decoder of a Transformer model. At each stage, the model has access to the already predicted words in the sentence. These models are often called auto regressive models. The pretraining of Decoder models usually includes predicting the next word in the sentence.

Table 2.1: NLP architectures table

Model	Examples	Tasks
Encoder	ALBERT,BERT,DistilBERT, ELECTRA,RoBERTa	Sentence classification, named entity recognition, extractive question answering
Decoder	CTRL,GPT, GPT-2, Transformer XL	Text Generation
Encoder-Decoder	BAPT,T5,Marion,mBart	Summarization, generative question answering

2.8 Masked Language Modeling (MLM)

As shown in figure 2.7 the process uniformly selects a percentage of the input tokens for possible replacement. 80% of the selected tokens are replaced with [MASK], 10% are left unchanged, and 10% is replaced by a randomly selected vocabulary token. In BERT's implementation, random masking and replacement are performed once in the prepossessing, and utilized for the whole the training, The data are duplicated so the mask is not the same for every training epoch. This technique is called static masking. Experiments in the RoBERTa paper reveal that this technique is inferior to dynamic masking. In dynamic masking, we apply masking in every batch. Dynamic masking not only uses less memory but leads to better performance.

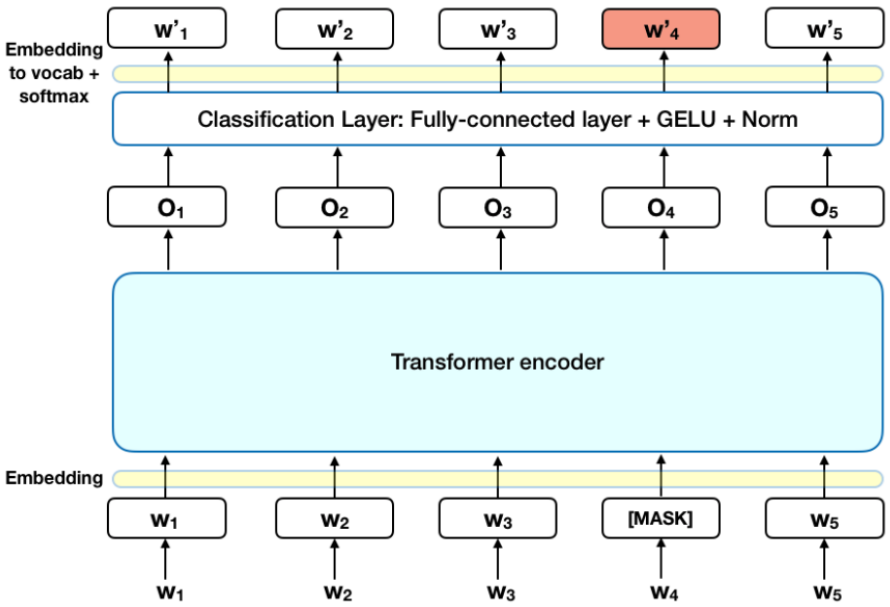

Figure 2.7: Masked Language Modeling

2.9 Bidirectional Encoder Representations for Transformers (BERT)

A Transformer based machine learning technique for Natural Language Processing (NLP) pretraining developed in 2018. The original BERT paper [18] introduced two models:

(1) BERT$_{BASE}$: 12 Encoders with 12 bidirectional self-attention heads. The total number of parameters sums up to 110 million

(2) BERT$_{LARGE}$: 24 Encoders with 16 bidirectional self-attention heads and 24 million. The total number of parameters sums up 340 to million.

Both models are pretrained from unlabeled data extracted from the BooksCorpus with 800M words and English Wikipedia with 2 500M words. BERT uses WordPiece embeddings with a 30 500 token vocabulary. BERT's significant performance is based on a technique called transfer learning. During transfer learning, the knowledge gained and rapid progress made from a source task is used to improve the learning and development of a new target task. This technique proved to be significantly useful in natural language tasks. There are two steps in transfer learning process as depicted in figure 2.8: pretraining and finetuning. During pretraining, the goal is that the model develops an understanding of the human language. The perspective of finetuning is to utilize the obtained knowledge in order to optimize the performance in a specific task.

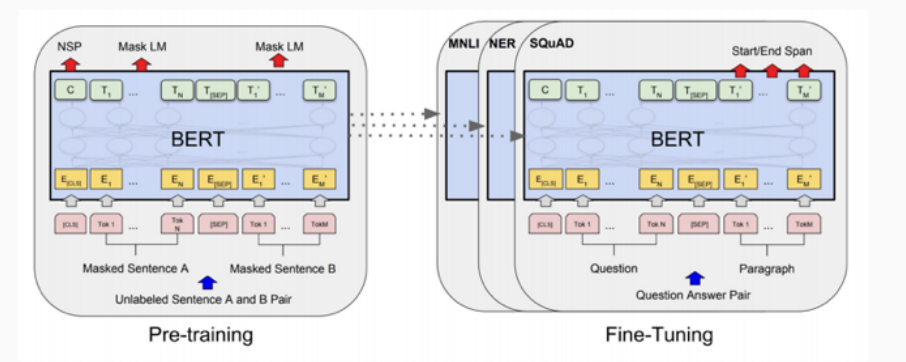

Figure 2.8: BERT

2.10 RoBERTa

RoBERTa[27] was introduced in 2019. In the original paper they demonstrated that BERT was significantly undertrained and proposed an improved recipe for training BERT models, which is called RoBERTa. More specifically this recipe includes:

- Amplify the vocabulary size to 50 264 tokens.
- Using GPT2 [21] Byte-Pair Encoding (BPE) [23] instead of word-piece.
- Removing the next sentence prediction objective [3].
- Training with batch size 8k instead of 256.
- Use x10 more Data.
- Dynamically changing the masking pattern applied to the training data.
- Training on sequences of length 512 instead of 128.
- Using FP16 mixed precision arithmetic.

This method led to significant performance gains and state of the art performance in GLUE, RACE, and SQuAD. In the following chapter we present the steps we made to pretrain our own RoBERTa model in Greek language.

3. PRETRAINING

The pretraining of models has yielded significant performance gains in the realm of Machine Learning. Especially state of the art models such as BERT [18], RoBERTa [27], XLNet [26], ALBERT [17], and T5 [22], among many others, have emerged due to pretraining. These methods, though they differ in design, share the same idea of leveraging a large amount of unlabeled text to build a general model of language understanding.

3.1 Pretraining dataset

We utilize the complete dataset accessible through the Nomothesi@ platform [7]. It consists of a huge amount of laws, announcements, and resolutions in the Greek language. The dataset spans a chronological range from 1990 to 2017, encompassing a substantial amount of information.

3.2 Inserting the data in a Hugging Face Dataset form

In order to preprocess our datasets more efficiently we need to insert them into a Hugging Face Dataset [4]. Hugging Face Datasets is a library for easily accessing and sharing datasets for Audio, Computer Vision, and Natural Language Processing (NLP) tasks.

Hugging Face Datasets allows us to load a dataset in a single line of code and use powerful data processing methods in order to quickly get a dataset ready for training for a deep learning model. Backed by the Apache Arrow format, our framework processes large datasets with zero copy, and reads without any memory constraints for optimal speed and efficiency. Hugging Face Datasets also features a deep integration with the Hugging Face Hub, allowing you to easily load and share a dataset with the wider machine learning community.

3.3 Text preprocessing

To preprocess and encode our text we are going to use Hugging Face Tokenizers. Hugging Face Tokenizers provide an implementation of today's most used tokenizers. Tokenizers are responsible the preprocessing of dataset. We also use them to decode *ids* that our model produces back into text. The tokenizers are separated into "*slow*" and "*fast*". Slow tokenizers are those written in python inside the Hugging Face Transformers library, while the fast versions are the ones provided by Hugging Face Tokenizers, which are written in Rust. The key in the fast tokenizers is parallelism.

In contrast to the original RoBERTa Paper [27], due to the distinctive attributes of our datasets, we need to apply preprocessing. Hugging Face provides the flexibility to customize the tokenizer according to individual requirements. So we built the tokenizer to support the following characteristics:

1. NFKD Normalization: Unicode normalization is the decomposition and composition of characters. Some unicode characters have the same appearance but multiple representations. For example, ”â” can be represented as one *code point* (U+00E2),

and two decomposed *code points* (U+0061) and (U+0302). We need to express all equal unicode characters in a single representation. In the beginning, we used NFD normalization as in GreekLegalBERT, but then we realized that NFD produced 2 different versions of the letter 'μ'.This could harm our model's performance given the fact that can be more instances like 'μ' that we don't know of. The solution was to use NFKD because K normalizations are more effective in removing formatting distinctions.

2. Remove accents: We remove accents due to the possibility of words in Greek having the same letters but differing in their accents. For instance, the term "νομοθεσία" is produced by the term law "νόμος". By eliminating the accents, we ensure that the text is standardized and accent variations do not hinder the accurate understanding and processing of the content.

3. Insert a space if it doesn't exist before every word: Because our encoder takes into consideration spaces to define if the token is the beginning or the end of a word we add a space at the beginning of every word before we proceed to the encoding and the training.

4. Detach punctuation: we separate text from punctuation by utilizing space.

In figure 3.1 we include the code we used to build our tokenizer.

```
[ ]  from tokenizers import (
         decoders,
         models,
         normalizers,
         pre_tokenizers,
         processors,
         trainers,
         Tokenizer,
     )
     Tokenizer(models.BPE())
```

```
<tokenizers.Tokenizer at 0x51bed00>
```

```
[ ]  tokenizer = Tokenizer(models.BPE())

     tokenizer.normalizer = normalizers.Sequence(
         [normalizers.NFKD(), normalizers.StripAccents()]
     )

     tokenizer.pre_tokenizer = pre_tokenizers.Sequence(
         [pre_tokenizers.Punctuation() , pre_tokenizers.ByteLevel()]
     )
```

```
tokenizer.pre_tokenizer.pre_tokenize_str("Let's test pre-tokenization!")
```

```
[('ĠLet', (0, 3)),
 ("Ġ'", (3, 4)),
 ('Ġs', (4, 5)),
 ('Ġtest', (5, 10)),
 ('Ġpre', (10, 14)),
 ('Ġ-', (14, 15)),
 ('Ġtokenization', (15, 27)),
 ('Ġ!', (27, 28))]
```

Figure 3.1: Building the Tokenizer

3.4 Byte Pair Encoder or BPE

In contrast to WordPiece [25] used by BERT, BPE [23] is an open-source algorithm. Byte-Pair Encoding (BPE) was initially developed as an algorithm to compress text, and then used for Tokenization when pre-training the GPT [8] model. It is used by a lot of Transformer models, including GPT, GPT-2, RoBERTa and BART. For non-ASCII characters, it gets completely unreadable, but it works nonetheless!

Early experiments revealed only slight differences between these encodings, with BPE achieving slightly worse end-task performance on some tasks.

Byte Pair Encoding (BPE) is a practical middle ground between character and word-level language modeling which effectively interpolates between word-level inputs for frequent symbol sequences and character level inputs for infrequent symbol sequences.

The algorithm can be described using the following two steps:

1. Start with an initial vocabulary that contains all the characters.
2. Find the most common combination of the current vocabulary and insert it in the vocabulary. Continue that step until you reach the vocabulary size.

In figure 3.2 we include the code we used to train our tokenizer.

3.5 The training procedure of the Tokenizer

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```

```
trainer = trainers.BpeTrainer( vocab_size=50264, min_frequency=2,
                special_tokens=['<s>', '<pad>', '</s>', '<unk>', '<mask>'])
```

```
[ ]  def get_training_corpus():
         dataset = raw_dataset["train"]
         for start_idx in range(0, len(dataset), 1000):
             samples = dataset[start_idx : start_idx + 1000]
             yield samples["text"]
```

```
tokenizer.train_from_iterator(get_training_corpus(), trainer=trainer)
```

Figure 3.2: Train the Tokenizer

During the training, the model creates a vocabulary of size 50 264. To reduce noise we add characters to our vocabulary that appear more than once because if a character appears only once on a 5GB dataset it is very likely to be corrupted. After 20 minutes our vocabulary is complete, and we are ready to proceed.

Our vocabulary consists of:

1. Whole words starting with space.
2. Punctuation.
3. Subwords occurring at the front of a word starting with space.
4. Subwords occurring at the end of a word or in isolation.
5. Individual characters.
6. Special tokens (<s>: beginning of sequence, </s>: end of sequence, <unk>: unknown token, <pad> Q: padding token, <mask> : masking token).

In figure 3.3 we use our tokenizer to convert our text to a sequence of numbers. At the following example, we see that some words with related meaning have close representation, but there is no guarantee that related word will be closer than non related ones.

```
print(tokenizer.encode( "παράνομος" ))
print(tokenizer.encode( "Παράνομος" ))
print(tokenizer.encode( "νόμος" ))
print(tokenizer.encode( "Νόμος" ))
print(tokenizer.encode( "νομοθέτης" ))
print(tokenizer.encode( "Νομοθέτης" ))
print(tokenizer.encode( "νομοθεσία" ))
print(tokenizer.encode( "Νομοθεσία" ))
print(tokenizer.encode( "νομοθετώ" ))
print(tokenizer.encode( "Νομοθετώ" ))
```

```
[402, 584, 554]
[992, 584, 554]
[10119]
[2285, 171, 554]
[1663, 488]
[3134, 488]
[3261]
[13440]
[1663, 619]
[3134, 619]
```

Figure 3.3: Tokenization example

3.6 Creating inputs for the training procedure

We separate each sequence into subsequences of 512 lengths. The last subsequence is less than 512 so we pad this subsequence in length 512. Padding is being applied dynamically during pretraining.

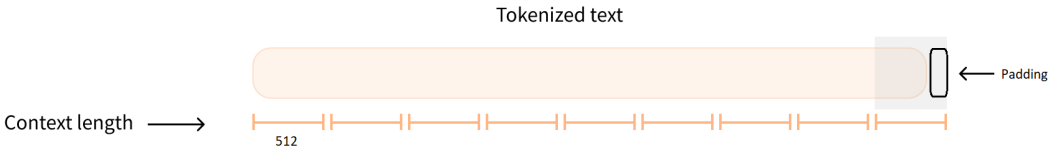

Figure 3.4: Create inputs for the training example

3.7 The original RoBERTa experiments

Even though RoBERTa [27] has a credible pretraining procedure, it is an expensive one. According to the BERT paper [13], longer sequences are disproportionately expensive because attention is quadratic to the sequence length. To speed up pretraining in their experiments, they pretrained the model with sequence length of 128 for 90% of the steps. Then, they train the rest 10% of the steps of sequence of 512 to learn the positional embeddings.

RoBERTa uses sequence length of 512 during the whole training procedure, which makes pretraining computationally expensive. As a result, due to the less powerful resources we have, it is unattainable to conduct RoBERTa experiments presented in figure 3.5. More specifically, training for 100k steps and batch size 8k would last approximately 240 days on our server.

Fortunately, we can still adjust our training to the relatively small dataset of 5GB we have and train our model for 100k steps as mentioned in RoBERTa paper and reduce our batch size to 1 024 . This process lasted 30 days on our server.

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Figure 3.5: Original RoBERTa experiments as they appear in [27]

Our server specifications are the following:

- An 8-core Intel® Core™ i7-9 700k, with 3.60GHz CPU frequency and 12.88Kb L3 cache.
- A Cannon Lake PCH Shared SRAM with a total of 32GB .
- An Nvidia GeForce® P0 with 11 016MB memory.
- Linux Ubuntu 18.04.6 LTS x86 64 OS.

3.8 Model configuration

RoBERTa [27] configuration follows BERT [13] configuration with a minor change in the embeddings. BERT consists of 110M parameters while RoBERTa consists of 125M parameters. The first and the last embedding of the model are randomly initialized. We also align our model's vocabulary size to tokenizer's vocabulary size. This adds approximately 15M additional parameters to our model because:

$$embedding\_matrix = vocab\_size * embedding\_space = 50\,264 * 768 - 30\,500 * 768 = 15\,178\,752$$

As a result our model consists of 125M parameters in total.

```
from transformers import RobertaConfig
from transformers import RobertaForMaskedLM
from transformers import RobertaTokenizerFast
from datasets import load_dataset
from transformers import Trainer, TrainingArguments
from transformers import DataCollatorForLanguageModeling

config = RobertaConfig(
    vocab_size=50264,
    max_position_embeddings=514,
    hidden_size=768,
    num_attention_heads=12,
    num_hidden_layers=12,
    type_vocab_size=1
)

model = RobertaForMaskedLM(config)
```

Figure 3.6: Model configuration

3.9 Training parameters

In this section, we will analyse the parameters utilized to train RoBERTa and the changes we made. We present the model parameters in figure 3.7. To train our model we use Hugging Face Trainers [10].

- **tqdm**: The progress bar that shows the percentage of training that the model has done.
- **output_dir**: The directory in which our trainer will save checkpoints.
- **batch_size**: The number of sequences that we are going to insert together to the model as an input. In our case this number is 8 because it is the threshold of our GPU capacity.
- **gradient_accumulation_steps**: The number of batches we insert into the model before we perform backpropagation. By increasing batch_size, the amount of GPU memory needed is increased exponentially. So, the gradient_accumulation_steps parameter function is to increase the batch_size while keeping the need for GPU memory stable. In our case gradient_accumulation_steps parameter is 128.

$$total\_batch\_size = batch\_size * gradient\_acumulationsteps = 1\,024$$

- **fp16**: Mixed floating point precision used to train the original model. This technique reduces the training memory while speeding up the training process.
- **learning_rate**: Due to the fact that we train for less time, we increase the peak learning rate from 6e-4 to 8e-4. At the beginning of our training process, learning rate is 1e-5. it is slowly increasing until the value of warmup steps when reaches 8e-4 then it decreases until the end of the training process.
- **adam_epsilon**: Epsilon remains 10^{-6} as in BERT and RoBERTa.
- **weight_decay**: Weight decay remains 0.01 as in BERT and RoBERTa.
- **adam_beta**: in RoBERTa beta1 remains the same as BERT. In contrast to beta2, it was found extremely sensitive to large batch sizes.
- **warmup_steps**: 6% of the total training steps.
- **hub_strategy**: The strategy we use to save the model and the tokenizer to the Hugging Face hub.
- **hub_token**: The token we use to push the model to the Hugging Face hub.

```
args = TrainingArguments(
    disable_tqdm = False,
    output_dir="Greek-Legal-Roberta_Uncased",
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    gradient_accumulation_steps=128,
    evaluation_strategy="steps",
    save_strategy="steps",
    save_total_limit=5,
    eval_steps=800,
    logging_steps=100,
    fp16=True,
    max_steps=100_000,
    weight_decay=0.01,
    learning_rate=8e-4,
    adam_epsilon=1e-6,
    adam_beta1=0.9,
    adam_beta2=0.98,
    warmup_steps=6_000,
    save_steps=53,
    hub_token="TOKEN"
)
```

Figure 3.7: Training arguments

3.10 Training process

At this stage, we are fully prepared to pretrain our model. Pretraining involves a process known as masked language modeling. In this process a certain percentage of our input is randomly masked and our model learns to predict the unknown words within this context. Through this procedure, the model develops a deep comprehension of the human language. Masking involves the following steps:

1. Pads the sequences to the maximum length, wich in our case is 512.
2. Randomly selects 15% of the input. Let's assume that the *i*th token has been chosen. There are three possible actions for this token: it will be replaced with <*mask*> with a probability of 80%, it will be substituted with a random token with a probability of 10%, or it will remain unchanged with a probability of 10%.
3. Creates labels: The labels are initialized with value -100 for non-masked tokens and the original token's value for the masked ones. Labels are -100 for non-masked tokens because we don't want them to contribute to the *cross-entropy loss*.

3.11 Model Testing

After completing the training, it is time to test if the model can stand up to the task that it was trained for, namely masked language modeling. So we provide our model with 5 rather challenging examples of text and we replace the words that the model will have to predict with <mask>. We present the 5 answers with the highest score.

According to our experiments which are presented in figure 3.8, it is evident that our model consistently generates reasonable answers. Additionally, our model produces the correct answer in 4 out of 5 times in the top 5 answers. Moreover, it produces also the version

```
======================Example1======================
text : Ορίζεται είδική υπηρεσία του <mask> γεωργίας.
Correct answer : υπουργείου
Model's answer : 1 :  τομεα , score : 0.4395993947982788
Model's answer : 2 :  γραφειου , score : 0.2374011129140854
Model's answer : 3 :  προγραμματος , score : 0.030760815367102623
Model's answer : 4 :  τμηματος , score : 0.024422641843557358
Model's answer : 5 :  Υπουργειου , score : 0.02435932494699955
======================Example2======================
text : Εγκριση παρεκκλίσεων στο υπο ανέγερση κτίριο της Υπηρεσίας <mask> Ερευνών Αρχηγείου ΕΛ.ΑΣ.
Correct answer : Εγκληματολογικων
Model's answer : 1 :  Οικονομικων , score : 0.3401985764503479
Model's answer : 2 :  Ειδικων , score : 0.1613190770149231
Model's answer : 3 :  Τεχνικων , score : 0.08202335983514786
Model's answer : 4 :  Εσωτερικων , score : 0.0703524649143219
Model's answer : 5 :  Πολιτικης , score : 0.04631315544247627
======================Example3======================
text : Κωδικοποίηση της <mask> για την κυβέρνηση και τα κυβερνητικά όργανα.
Correct answer : νομοθεσίας
Model's answer : 1 :  νομοθεσιας , score : 0.9262252449989319
Model's answer : 2 :  Νομοθεσιας , score : 0.07356727868318558
Model's answer : 3 :  νομοθεσια , score : 0.0002002511319005862
Model's answer : 4 :  συνθηκης , score : 9.048104629982845e-07
Model's answer : 5 :  Νομοθεσια , score : 8.852931614455883e-07
======================Example4======================
text : Περί εξαιρέσεως από τις <mask> των νομαρχών μερικών αντικειμένων
Correct answer : αρμοδιότητες
Model's answer : 1 :  αρμοδιοτητες , score : 0.8470842838287354
Model's answer : 2 :  αρμοδιοτητας , score : 0.0628998801112175
Model's answer : 3 :  αρμοδιοτητος , score : 0.05345875024795532
Model's answer : 4 :  κατα , score : 0.015101701021194458
Model's answer : 5 :  αρμοδιοτητα , score : 0.007082927506417036
======================Example5======================
text : Καθορισμός τρόπου αξιολόγησης της επιμελείς των υπαλλήλων που παρακολουθούν προγράμματα επιμόρφωσης και <mask> .
Correct answer : εξειδίκευσης
Model's answer : 1 :  καταρτισης , score : 0.42756062746047974
Model's answer : 2 :  αξιολογησης , score : 0.12421643733978271
Model's answer : 3 :  μετεκπαιδευσης , score : 0.08208373188972473
Model's answer : 4 :  ενημερωσης , score : 0.046267617493867874
Model's answer : 5 :  εξειδικευσης , score : 0.04247415438294411
```

Figure 3.8: Model testing on masked language modeling

of the correct answer with the first letter capitalized, indicating its understanding of the equivalent meanings between the two variations. The above fact is crucial to our model performance. This is due to the fact that a lot of words in Greek such as names, geopolitical entities, titles of honor, and much more are written with the first letter capitalized. But why the model did not produce the second answer ”Εγκλημματολογικων” which means Forensics in the answer demonstrated in figure 3.9? This can be attributed to the fact that the word is being split into two tokens during the encoding process. We try overcome this obstacle by applying 2 masks.

```
unmasker('Εγκριση παρεκκλίσεων στο υπο ανέγερση κτίριο της Υπηρεσίας <mask> <mask> Ερευνών Αρχηγείου ΕΛ.ΑΣ.')
```

```
[[{'score': 0.27278175950050354,
   'token': 4815,
   'token_str': ' Προγραμματισμου',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας Προγραμματισμου<mask> Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.11765486001968384,
   'token': 3532,
   'token_str': ' Κεντρου',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας Κεντρου<mask> Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.07667404413223267,
   'token': 1846,
   'token_str': ' Εκπαιδευσης',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας Εκπαιδευσης<mask> Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.05240662023425102,
   'token': 35328,
   'token_str': ' Μηχανογραφησης',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας Μηχανογραφησης<mask> Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.03583092987537384,
   'token': 24616,
   'token_str': ' Περιοδ',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας Περιοδ<mask> Ερευνων Αρχηγειου ΕΛ. ΑΣ.'}],
 [{'score': 0.7701647281646729,
   'token': 37870,
   'token_str': 'ματολογικων',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας<mask>ματολογικων Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.14580486714839935,
   'token': 219,
   'token_str': ' και',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας<mask> και Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.01859731785953045,
   'token': 371,
   'token_str': 'ικων',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας<mask>ικων Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.011692304164171219,
   'token': 448,
   'token_str': 'τικων',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας<mask>τικων Ερευνων Αρχηγειου ΕΛ. ΑΣ.'},
  {'score': 0.00852262880653143,
   'token': 902,
   'token_str': ' Οικονομικων',
   'sequence': ' Εγκριση παρεκκλισεων στο υπο ανεγερση κτιριο της Υπηρεσιας<mask> Οικονομικων Ερευνων Αρχηγειου ΕΛ. ΑΣ.'}]]
```

Figure 3.9: Model questioning

As observed in figure 3.9, the second subtoken of this word ”Εγκλημματολογικων” obtains a score of 0.77. Unfortunately, predicting words splitted in more than one token during tokenization is not supported in the Transformers pipeline. For future research it would be interesting to research how we can train a model and utilize it in order to predict whole words within the sentence, independent of how many tokens the word is being split into. A process like this can be used also in pretraining, and contribute to the model’s intuition of the human language.

After validating our pretraining process it is time to evaluate it. More specifically, we are going to evaluate out model’s performance on two downstream tacks which are Named Entity Recognition and Multiclass classification, in order to see if our pretraining method leads to a better performance.

4. EXPERIMENTS

We provide a comparative evaluation focusing on the assessment of our model in comparison to existing state of the art Greek models GreekBERT [16] and GreekLegalBERT [15]. In tables 4.3 and 4.5, we present our results. We observe that GreekLegalRoBERTa provides improvement over the originally reported GreekBERT and GreekLegalBERT.

4.1 The models we experiment with

In this chapter we present all the necessary details of the models we experiment with.

GreekBERT [16]: A monolingual version of BERT, trained solely on modern Greek, achieving state of the art results in most of the Greek NLP tasks. GreekBERT was pretrained on 29GB of text from a corpus consisting of the Greek part of Wikipedia, the Greek part of the European Parliament Proceedings Parallel Corpus [14] and OSCAR [19].

GreekLegalBERT [15]: A monolingual legal version of BERT. GreekLegalBERT and our model were pretrained on the same dataset 3.1. Despite the smaller dataset, this model managed to exceed the performance of GreekBERT in several legal tasks.

Given the fact that training is equal to the batch size multiplied by the steps, it is evident that the other models have undergone 2.5 times more training compared to our model. Moreover, GreekBERT uses 6 times more data than the other models. Nonetheless GreekLegalBERT and GreekLegalRoBERTa are specifically trained on legal text. As we can see in the table below given the fact that :

$$total\_training = batch\_size * training\_steps$$

we have :

$$BERT\_models = 2.5 * total\_training\_of\_our\_model$$

Moreover, GreekBERT uses x6 more data than the other models but GreekLegalBERT and GreekLegalRoBERTa are specifically trained in legal text.

Table 4.1: NER Dataset Statistics

Model	Size of data	Training Steps	Batch Size	Total Training
GreekBERT	30GB	1M	256	256M
GreekLegalBERT	5GB	1M	256	256M
GreekLegalRoBERTa	5GB	100K	1024	102.4M

4.2 Greek Legal NER

Dataset

The dataset contains 254 daily issues for classes of the Greek Government Gazette over the period 2000-2017. Every issue contains multiple legal acts. This dataset is focusing on 7 entity types (legislation references, geopolitical entities, national locations, unknown locations, public locations, organizations, and facilities). The dataset was available in

Inside Outside Beginning (IOB), which is a common tagging format for tagging tokens for NER. In figure 4.1, we include a part of the dataset as an example. It consists of 35 411 instances and it is divided into 3 main parts: *train* 67.5%, *validation* 17.5%, and *test* 15%.

- Train Set
  The sample of data that our model will have to acquire the knowledge from. During training, the model passed through the data in order to learn their characteristics. During Hyperparameter tuning we train our model multiple times for different parameter combinations.
- Validation Set
  The sample of data that we will use to evaluate our model performance during the hyperparameter tuning in order to find which hyperparameters are best for our model. We can not use the training set to tune this hyperparameters because we need to provide a performance estimation for a dataset that our model hasn't seen.
- Test Set:
  The sample of data we use to provide an unbiased evaluation of our model at the end of the fintuning. We use this set to avoid a phenomenon called overfitting on the validation set. When this phenomenon occurs, the models perform great on the validation set, because we did hyperparameter tuning on the validation set, but this performance does not generalize. So we use a new dataset to make the performance of the model generalize on datasets that the model hasn't seen.

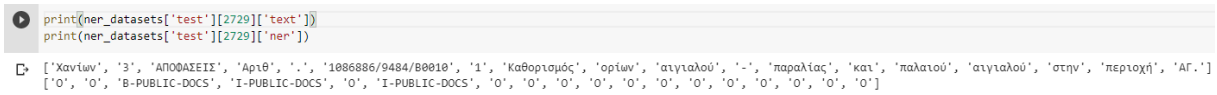

Figure 4.1: NER presentation

Preprocessing for NER

The first step of preprocessing is the encoding. During the encoding, many words are divided into multiple tokens, but the labels remain stable. To address this issue we have to align the labels and the encodings. To do this we apply 3 types of transformation. Consider the example that a word is divided into 3 tokens.

- If the Word label is B-XXX the label will be converted to B-XXX I-XXX I-XXX.
- If the Word label is I-XXX the label will be converted to I-XXX I-XXX I-XXX.
- If the Word label is O the label will be converted to O O O.

Hyperparameter tuning

In order to find the optimal hyperparameters for each task we utilize grid search [9]. In this method, we train our models using different hyperparameter combinations and we keep the one that achieves the best micro F1 in the validation set. For each model we experiment with the following hyperparameters:

- epochs (from 1 to 20).
- batch size (8,16).
- learning rate (2e-5, 3e-5, 5e-5).

The table below 4.2 presents the hyperparameter combinations at which the models achieved their best results.

Table 4.2: Model performance on finetunig

Model	learning rate	epochs	batch_size
GreekBERT	5e-5	3	8
GreekLegalBERT	3e-5	3	8
GreekLegalRoBERTa	5e-5	6	8

Results

We perform 5 runs of training and model evaluation per model using 5 different *seeds*. Finally, we provide the classification report, including the mean and the standard deviation of the 5 experiments 4.3, for each model's performance on the *test* set. In order to perform a comparative evaluation, we will highlight the best F1 performance evaluation for each entity.

Table 4.3: Model performance on NER (LEGISLATION REFERENCES: LR, LOCATION NATIONAL: LN, LOCATION UNKNOWN: LU, PUBLIC DOCUMENTS: PD)

	GreekBERT			Greek Legal BERT			Greek Legal RoBERTa		
	precision	**recall**	**F1**	**precision**	**recall**	**F1**	**precision**	**recall**	**F1**
FACILITY	32 (2%)	28 (4%)	29 (3%)	33 (4%)	26 (3%)	29 (3%)	33 (3%)	28 (3%)	**30 (2%)**
GPE	79 (1%)	72 (1%)	**75 (1%)**	77 (1%)	71 (1%)	74 (1%)	84 (1%)	80 (0%)	**75 (0%)**
LR	84 (1%)	81 (1%)	**82 (0%)**	84 (1%)	81 (0%)	**82 (0%)**	84 (1%)	80 (1%)	**82 (1%)**
LN	100 (0%)	80 (13%)	**88 (11%)**	90 (22%)	29 (12%)	43 (16%)	100 (0%)	40 (20%)	55 (20%)
LU	76 (1%)	70 (2%)	**73 (1%)**	73 (2%)	68 (3%)	71 (2%)	74 (1%)	68 (1%)	71 (1%)
ORG	78 (1%)	71 (1%)	74 (1%)	80 (1%)	71 (1%)	76 (1%)	82 (1%)	73 (1%)	**77 (1%)**
PERSON	89 (1%)	81 (1%)	85 (1%)	91 (1%)	83 (2%)	86 (1%)	91 (2%)	83 (1%)	**87 (1%)**
PD	70 (1%)	66 (2%)	**68 (1%)**	70 (2%)	67 (1%)	**68 (2%)**	69 (1%)	66 (2%)	**68 (0%)**
micro	79	73	76	79	73	76	80	73	**77**
macro	76	68	**72**	75	62	66	76	64	68
weighted	79	73	76	79	73	76	80	73	**77**

Based on our findings shown in figure 4.3, it is evident that our model, despite undergoing significantly less pretraining, outperforms GreekLegalBERT in every entity. Also, our model emerges with the best score in 6 out of 8 categories while GreekBERT emerges with the highest score in 5 out of 8. Furthermore, our model demonstrates superior performance in *micro* and *weighted* average, indicating that our model performs better than the others in the majority of cases. Unfortunately, the model does not outperform GreekBERT in the macro average. This is due to the fact that even though the model has very competitive accuracy in the location entities, the recall is very low. Consequently, in case there is a location entity, the model can identify it with high accuracy, but it is also misclassifying entities as locations that are not. Another reason can be that the model has difficulty discretizing between the different types of locations. This imbalance can be attributed to the nature of the data GreekBERT was trained on. It seems that the data GreekBERT was trained on a dataset that contained more instances of locations. For this reason, it would be crucial to develop a new version of RoBERTa trained in both our dataset and the dataset utilized for training GreekBERT.

4.3 Greek Legal Code

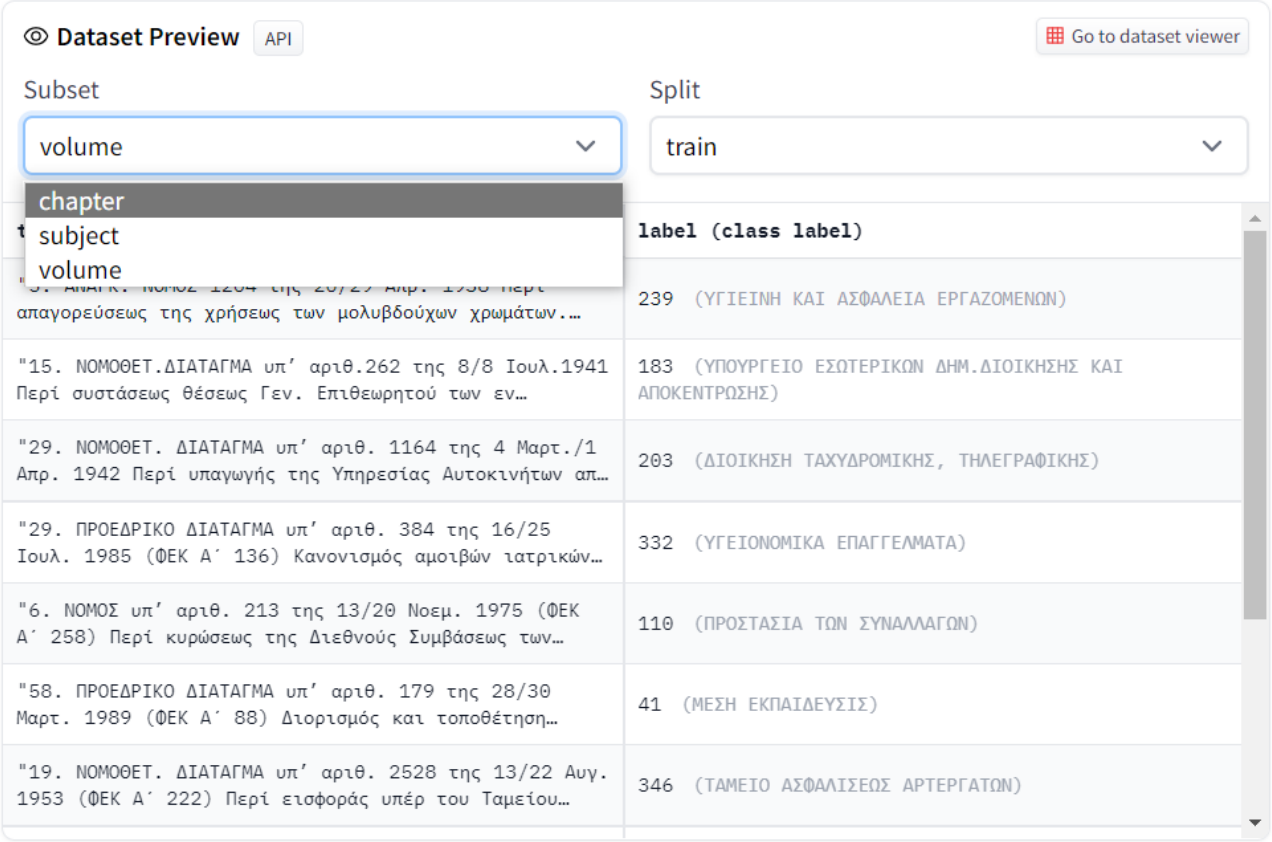

Figure 4.2: GLC Dataset Hugging-Face as presented in Hugging-Face website

Lastly, we will conduct experiments using Greek Legal Code (GLC) [20] which is available in Hugging Face. Figure 4.2 presents the preview of the dataset as it appears in the GLC dataset page. This study introduces a new dataset of legal context and proves that in this dataset GreekLegalBERT outperforms all the previous Greek and multilingual models. In table 4.5 we show that our model outperforms all the previous ones including GreekLegalBERT

Dataset

The dataset is a thorough catalog of Greek legislation. It includes Laws, Royal and Presidential Decrees, Regulations, and Decisions, retrieved from the Official Government Gazette. The catalog is structured into thematic topics making the data ideal for multi-label classification. It consists of 47 legislative *volumes* and each *volume* corresponds to a main thematic topic. Each *volume* is divided into thematic subcategories which are called *chapters* and subsequently, each *chapter* breaks down into *subjects*. The total number of *chapters* is 389 while the total number of *subjects* is 2285. The dataset is divided in 3 parts shown in 4.4 which are train, test and validation.

Results

As previously we perform 5 runs of training and model evaluation on the test set per model using 5 different seeds. Finally, we present the mean of *micro* F1, precision, and recall of the performance evaluation. In our experiments shown in 4.5, it is evident that our model exhibits superior performance when compared to all other models in volume, chapter, and subject classification.

Table 4.4: GLC Dataset Statistics

Dataset	percentage	instances
Train	60%	28536
Test	20%	9516
Validation	20%	9511

Table 4.5: GLC Dataset configuration

	Volume			**Chapter**			**Subject**		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
GreekBERT	89.84	89.84	89.84	84.87	84.87	84.87	80.59	80.59	80.59
GreekLegalBERT	90.51	90.51	90.51	85.45	85.45	85.45	81.43	81.43	81.43
GreekLegalRoBERTa	**91.10**	**91.10**	**91.10**	**85.77**	**85.77**	**85.77**	**82.29**	**82.29**	**82.29**

In the previous sections we demonstrated that our model outperforms GreekLegalBERT and GreekBERT in GreekLegalNER (figure 4.3) and GreekLegalCode (figure 4.5). Consequently, our results suggest that the dataset and pretraining method we utilize effectively enhance the model's performance on legal context. Nevertheless, given the fact that our model is significantly undertrained, there is a prominent possibility that further pretraining can lead to an improvement in model performance.

5. CONCLUSION AND FUTURE WORK

In this work, we introduced a new language model pretrained solely on legal text. We pretrained our model on a single GPU, and our findings demonstrate its superior performance compared to the state of the art Greek NLP models. For future research, we plan to train our model on a GPU cluster, utilizing a significantly larger batch size for an extended number of epochs. These enhancements aim to further improve the model's performance and achieve even better results.

Additionally, we are going to train our model using both natural language and legal context to investigate the effectiveness of this approach. This research aims to determine whether incorporating both domains is a beneficial practice for improving the model's performance and overall understanding.

ABBREVIATIONS - ACRONYMS

AI	Artificial Intelligence
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly optimized BERT pretraining approach
GPT	Generative pre-training
BART	Bidirectional and Auto-Regressive Transformers
TAPT	Task Adaptive pre-training
DAPT	Domain Adaptive pre-training
DNN	Deep Neural Networks
ANN	Artificial Neural Networks
FFANN	Feed Forward Artificial Neural Networks
RNNs	Recurrent Neural Networks
LSTM	Long Short Term Memory
BiLSTM	Bidirectional Long Short Term Memory
MLM	Masked Language Modeling
BPE	Byte-Pair Encoding
IOB	Inside Outside Beginning
NER	Named Entity Recognition
PoS	Part of Speech tagging
GLUE	General Language Understanding Evaluation benchmark
RACE	Large-scale Reading Comprehension Dataset From Examinations
SQuAD	Stanford Question Answering Dataset
FP	Floating Point
NFD	Normalization Form Canonical Decomposition
NFKD	Normalization Form Compatibility Decomposition

TPU	Tensor Processing Unit
GPU	Graphics Processing Unit
GLC	Greek Legal Code
CPU	Central Processing Unit
RAM	Random Access Memory
SSD	Solid State Drive
GPE	GeoPolitical Entity
LR	Legal References
LN	Location National
LU	Location Unknown
ORG	Organization
PD	Public Documents
AVG	Average

BIBLIOGRAPHY

[1] Artificial neural network. https://en.wikipedia.org/wiki/Artificial_neural_network.

[2] Attention (machine learning). https://en.wikipedia.org/wiki/Attention_(machine_learning).

[3] Bert for next sentence prediction. https://towardsdatascience.com/bert-for-next-sentence-prediction-466b67f8226f#:~:text=N%20ext%20sentence%20prediction%20%28NSP%29%20is%20one-half%20of,teaches%20BERT%20to%20understand%20longer-term%20dependencies%20across%20sentences.

[4] Hugging face datasets. https://huggingface.co/docs/datasets/index.

[5] Natural language processing. https://en.wikipedia.org/wiki/Natural_language_processing.

[6] Perceptron. https://en.wikipedia.org/wiki/Perceptron.

[7] Ilias Chalkidis, Charalampos Nikolaou, Panagiotis Soursos, and Manolis Koubarakis. Modeling and querying greek legislation using semantic web technologies. In Eva Blomqvist, Diana Maynard, Aldo Gangemi, Rinke Hoekstra, Pascal Hitzler, and Olaf Hartig, editors, *The Semantic Web*, pages 591–606, Cham, 2017. Springer International Publishing.

[8] Leonhard Hennig Christoph Alt, Marc Hübner. Fine-tuning pre-trained transformer language models to distantly supervised relation extraction. 2018.

[9] Hugging Face. Hyperparameter search with transformers and ray tune. https://huggingface.co/blog/ray-tune.

[10] Hugging Face. Trainer. https://huggingface.co/docs/transformers/main/en/main_classes/trainer#transformers.Trainer.

[11] Santiago Fernandez Justin Bayer Daan Wierstra Julian Togelius Faustino Gomez Matteo Gagliolo Felix Gers, Fred Cummins and Alex Graves. Feedforward neural networks. https://brilliant.org/wiki/feedforward-neural-networks/.

[12] Santiago Fernandez Justin Bayer Daan Wierstra Julian Togelius Faustino Gomez Matteo Gagliolo Felix Gers, Fred Cummins and Alex Graves. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[14] Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of Machine Translation Summit X: Papers*, pages 79–86, Phuket, Thailand, September 13-15 2005.

[15] Athinaios Konstaninos. Named entity recognition using a novel linguistic model for greek legal corpora based on bert model. 2020. Dept. Informatics and Telecommunication, National and Kapodistrian University of Athens.

[16] John Koutsikakis, Ilias Chalkidis, Prodromos Malakasiotis, and Ion Androutsopoulos. GREEK-BERT: the greeks visiting sesame street. In Constantine D. Spyropoulos, Iraklis Varlamis, Ion Androutsopoulos, and Prodromos Malakasiotis, editors, *SETN 2020: 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, September 2-4, 2020*, pages 110–117. ACM, 2020.

[17] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019.

[18] Naman Goyal Marjan Ghazvininejad Abdelrahman Mohamed Omer Levy Ves Stoyanov Luke Zettlemoyer Mike Lewis, Yinhan Liu. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2019.

[19] Pedro Javier Ortiz Suárez, Laurent Romary, and Benoît Sagot. A monolingual approach to contextualized word embeddings for mid-resource languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1703–1714, Online, July 2020. Association for Computational Linguistics.

[20] Christos Papaloukas, Ilias Chalkidis, Konstantinos Athinaios, Despina Pantazi, and Manolis Koubarakis. Multi-granular legal topic classification on Greek legislation. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 63–75, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2018.

[22] 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. *CoRR*, abs/1910.10683, 2019.

[23] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, August 2016. Association for Computational Linguistics.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[25] Yang Song Dave Dopson Denny Zhou Xinying Song, Alex Salcianu. Fast wordpiece tokenization. 2021.

[26] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

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