



**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

**SCHOOL OF SCIENCES  
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS**

**BSc THESIS**

## **NLP Tasks with GreekLegalBERT v2**

**Alexandra G. Apostolopoulou  
Spyridon A. Briakos**

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

**ATHENS**

**DECEMBER 2021**



**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ  
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

**Προβλήματα Επεξεργασίας Φυσικής Γλώσσας με την  
δεύτερη έκδοση του Ελληνικού Νομικού BERT μοντέλου**

**Αλεξάνδρα Γ. Αποστολοπούλου  
Σπυρίδων Α. Μπριάκος**

**Επιβλέποντες: Μανόλης Κουμπάρκης, Καθηγητής  
Δέσποινα - Αθανασία Πανταζή, Υποψήφια Διδάκτωρ**

**ΑΘΗΝΑ**

**ΔΕΚΕΜΒΡΙΟΣ 2021**

## **BSc THESIS**

NLP Tasks with GreekLegalBERT v2

**Alexandra G. Apostolopoulou**

**S.N.:** 1115001700005

**Spyridon A. Briakos**

**S.N.:** 1115001700101

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

## **ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ**

Προβλήματα Επεξεργασίας Φυσικής Γλώσσας με την δεύτερη έκδοση του Ελληνικού Νομικού BERT μοντέλου

**Αλεξάνδρα Γ. Αποστολοπούλου**

**A.M.: 1115001700005**

**Σπυρίδων Α. Μπριάκος**

**A.M.: 1115001700101**

**ΕΠΙΒΛΕΠΟΝΤΕΣ:** **Μανόλης Κουμπάρκης, Καθηγητής**  
**Δέσποινα - Αθανασία Πανταζή, Υποψήφια Διδάκτωρ**

## ABSTRACT

In recent years, Legal Artificial Intelligence (LegalAI) has drawn increasing attention rapidly from both AI researchers and legal professionals [29]. This is due to the fact that LegalAI benefits the legal system for liberating legal professionals from a maze of paperwork. LegalAI focuses on applying artificial intelligence technologies, particularly Natural Language Processing, to benefit tasks in legal domain.

As part of this endeavor, BERT has started to be employed in legal domain, mostly in English language, for tasks such as legal judgement prediction and violation prediction [12], as it is well recognized for its remarkable performance in a variety of NLP tasks. As far as we can discern, there are two unique models in the Greek NLP era: the general-purpose Greek-BERT [21] model and the specific-domain Greek-Legal-BERT-v1 [20] model. In this thesis, we focus on the generation and representation of the second version of Greek-Legal-BERT, namely GreekLegalBERT v2, which was provided with more Legal Data than the first version. So, our primary goal is to increase the size of Greek NLP models. The current thesis' next goal is to compare the three distinct Greek NLP models, based on BERT [15] model, between different downstream NLP tasks, notably in Named Entity Recognition, Natural Language Inference and Multiclass Classification on Raptarchis [13] dataset. Therefore, we consciously choose to begin our study with BERT's Architecture and the tasks it has been pretrained on. Having a thorough knowledge of the BERT model, we continue with break down Greek-BERT and Greek-Legal-BERT-v1 to highlight the distinctions between models. Then, using more Legal Data than Greek-Legal-BERT-v1, we proceed to train our own comparable GreekLegalBERT v2 model, with aim of maximizing its performance.

Finally, we assessed the three previously mentioned NLP models on the three previously mentioned NLP tasks and presented the results. Precision, Recall and F1 score were the metrics we used in our evaluation. We selected these metrics so that we could compare them directly to previous models evaluated on the same dataset. In terms of NER, our model achieved the greatest performance with a slight difference among the two remaining models. As far as NLI is concerned, our model outperforms Greek-Legal-BERT-v1 in all three categories, but it didn't manage to surpass the scores of Greek-BERT. Eventually, in case of Multiclass Classification on Raptarchis dataset, our model surpassed in the first two categories, namely Volume and Chapter, however in the last category, Subject, Greek-Legal-BERT-v1 was the unique winner among all models.

**SUBJECT AREA:** Natural Language Processing

**KEYWORDS:** BERT, Named Entity Recognition, Natural Language Inference, Legal Data, Compare Greek Models

## ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, η Τεχνητή Νοημοσύνη με Νομικά Δεδομένα προσελκύει ολοένα και μεγαλύτερη προσοχή τόσο από τους ερευνητές της Τεχνητής Νοημοσύνης όσο και από τους επαγγελματίες νομικούς. Αυτό οφείλεται στο γεγονός ότι η Τεχνητή Νοημοσύνη με Νομικά Δεδομένα ωφελεί το νομικό σύστημα μέσω της απελευθέρωσης των επαγγελματιών νομικών από έναν επίπονο λαβύρινθο γραφειοκρατίας. Η Τεχνητή Νοημοσύνη με Νομικά Δεδομένα επικεντρώνεται στην εφαρμογή τεχνολογιών τεχνητής νοημοσύνης, ιδιαίτερα στην Επεξεργασία Φυσικής Γλώσσας, προς όφελος εργασιών σε νομικό τομέα.

Ως μέρος αυτής της προσπάθειας, το μοντέλο BERT έχει αρχίσει να χρησιμοποιείται στο νομικό τομέα, κυρίως στην Αγγλική γλώσσα, για εργασίες όπως η πρόβλεψη νομικής κρίσης και η πρόβλεψη παραβίασης [12], καθώς είναι ευρέως αναγνωρισμένο για την αξιοσημείωτη απόδοση του σε μια πληθώρα διεργασιών Επεξεργασίας Φυσικής Γλώσσας. Από όσα μπορούμε να γνωρίζουμε, υπάρχουν δύο μοναδικά μοντέλα που αφορούν την Ελληνική γλώσσα: το μοντέλο Greek-BERT [21] και το μοντέλο Greek-Legal-BERT-v1 [20]. Σε αυτήν τη διατριβή, εστιάζουμε στη δημιουργία και την παρουσίαση της δεύτερης έκδοσης του Greek-Legal-BERT, δηλαδή του GreekLegalBERT v2, το οποίο δημιουργείται με περισσότερα Νομικά Δεδομένα από την πρώτη έκδοση. Έτσι, πρωταρχικός μας στόχος είναι η αύξηση του μεγέθους των Ελληνικών μοντέλων Επεξεργασίας Φυσικής Γλώσσας. Ο επόμενος στόχος της τρέχουσας διατριβής είναι η σύγκριση των τριών Ελληνικών μοντέλων, που βασίζονται στο μοντέλο BERT [15], μεταξύ διαφόρων διεργασιών Επεξεργασίας Φυσικής Γλώσσας, δηλαδή στην Αναγνώριση Ονοματισμένης Οντότητας, την Εξαγωγή Φυσικής Γλώσσας και την Πολυεπίπεδη Ταξινόμηση στο σύνολο δεδομένων Raptarchis [13]. Έπειτα, αποφασίσαμε συνειδητά να ξεκινήσουμε τη μελέτη μας για την αρχιτεκτονική του BERT και τις θεμελιώδεις διεργασίες στις οποίες έχει προεκπαιδευτεί. Έχοντας μια πλήρη γνώση του μοντέλου BERT, συνεχίζουμε με τη περαιτέρω ανάλυση των Greek-BERT και Greek-Legal-BERT-v1 για να επισημάνουμε τις διαφορές μεταξύ των μοντέλων. Εν συνέχεια, χρησιμοποιώντας περισσότερα Νομικά Δεδομένα από το Greek-Legal-BERT-v1, προχωρούμε στην εκπαίδευση του δικού μας μοντέλου GreekLegalBERT v2, με απώτερο στόχο τη μεγιστοποίηση της απόδοσής του.

Τέλος, αξιολογούμε τα τρία προαναφερθέντα μοντέλα για τις τρεις προαναφερθείσες διεργασίες Επεξεργασίας Φυσικής Γλώσσας και παρουσιάζουμε τα αποτελέσματα. Η ακρίβεια, η ανάκληση και η βαθμολογία F1 ήταν οι μετρικές που χρησιμοποιήσαμε στην αξιολόγησή μας. Επιλέξαμε αυτές τις μετρικές, ούτως ώστε να μπορούμε να τα συγκρίνουμε άμεσα με προηγούμενα μοντέλα που αξιολογήθηκαν στο ίδιο σύνολο δεδομένων. Όσον αφορά το NER, το μοντέλο μας πέτυχε τη μεγαλύτερη απόδοση με μια μικρή διαφορά μεταξύ των δύο υπολοίπων μοντέλων. Όσον αφορά το NLI, το μοντέλο μας ξεπερνά το Greek-Legal-BERT-v1 και στις τρεις κατηγορίες, αλλά δεν κατάφερε να ξεπεράσει τις βαθμολογίες του Greek-BERT. Τέλος, στην περίπτωση της Πολυεπίπεδης Ταξινόμησης στο σύνολο δεδομένων Raptarchis, το μοντέλο μας ξεπέρασε στις δύο πρώτες κατηγορίες, ωστόσο στην τελευταία κατηγορία, το Greek-Legal-BERT-v1 ήταν ο μοναδικός νικητής μεταξύ όλων των μοντέλων.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ:** Επεξεργασία Φυσικής Γλώσσας

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:** Μοντέλο BERT, Αναγνώριση Ονοματισμένων Οντοτήτων, Συμπέρασμα Φυσικής Γλώσσας, Νομικά Δεδομένα, Σύγκριση Ελληνικών Μοντέλων

*Thanks to our parents for never losing hope in us.  
Without you, we wouldn't have made it this far.  
We are extremely grateful to all.*

## **ACKNOWLEDGEMENTS**

For the completion of this Thesis, we would like to thank our supervisors, namely the PhD candidate Despina-Athanasia Pantazi and our professor Manolis Koubarakis, for their cooperation and their valuable contribution and guidance throughout each stage of our thesis' planning and development. We are incredibly appreciative for the opportunity to speak about such an intriguing and major topic.



# CONTENTS

<b>1. INTRODUCTION</b>	<b>14</b>
<b>2. BACKGROUND AND RELATED WORK</b>	<b>16</b>
<b>2.1 BERT Model</b>	<b>16</b>
2.1.1 What is BERT?	16
2.1.1.1 B: Bi-directional	16
2.1.1.2 ER: Encoder Representations	17
2.1.1.3 T: Transformers	17
2.1.2 BERT's Architecture	18
2.1.3 How does it work?	19
2.1.3.1 Input	19
2.1.3.2 Output	20
2.1.4 Tasks that BERT has been pre-trained on	20
2.1.4.1 Masked Language Modeling	21
2.1.4.2 Next Sentence Prediction	21
<b>2.2 Comparison with other BERT Models</b>	<b>22</b>
2.2.1 Greek-BERT	22
2.2.2 Greek-Legal-BERT-v1	22
<b>3. PRETRAIN OF GREEKLEGALBERT V2 MODEL</b>	<b>23</b>
<b>3.1 Datasets and Preprocessing</b>	<b>23</b>
3.1.1 Raptarchis	23
3.1.2 Nomothesi@ Platform	24
3.1.3 EuroParl	26
3.1.4 EUR-LEX	26
3.1.5 Hellenic Parliament Sessions	30
3.1.6 Preprocessing	31
3.1.7 Tokenization	32
3.1.8 Generate Vocabulary	33
<b>3.2 Google Cloud Setup</b>	<b>33</b>
3.2.1 Why Choose Google Cloud Platform?	33
3.2.2 TPU Quotas	34
3.2.3 Create Google Compute Engine (GCE)	34
3.2.4 Create Tensorflow Records (TFRecords)	34
3.2.5 Create Google Cloud Storage (GCS)	35
3.2.6 Create Tensor Processing Unit (TPU) Node	36
3.2.7 Prepare Virtual Machine (VM) Environment	36
<b>3.3 Model Pretraining</b>	<b>37</b>
3.3.1 Pretraining Parameters	37
3.3.2 Pretraining Procedure	37
3.3.3 Save model	38

3.3.4	Upload model . . . . .	39
<b>4.</b>	<b>TASKS OF GREEKLEGALBERT V2 MODEL</b>	<b>41</b>
<b>4.1</b>	<b>Named Entity Recognition . . . . .</b>	<b>41</b>
4.1.1	Dataset . . . . .	41
4.1.2	Fine Tuning . . . . .	43
<b>4.2</b>	<b>Natural Language Inference . . . . .</b>	<b>44</b>
4.2.1	Dataset . . . . .	45
4.2.2	Fine Tuning . . . . .	46
<b>4.3</b>	<b>Multiclass Classification on Raptarchis . . . . .</b>	<b>48</b>
4.3.1	Dataset . . . . .	48
4.3.2	Fine Tuning . . . . .	49
<b>4.4</b>	<b>Performance on Masked Language Modeling (MLM) . . . . .</b>	<b>49</b>
<b>4.5</b>	<b>Results . . . . .</b>	<b>51</b>
4.5.1	Named Entity Recognition . . . . .	52
4.5.2	Natural Language Inference . . . . .	53
4.5.3	Multiclass Classification on Raptarchis . . . . .	53
<b>5.</b>	<b>CONCLUSIONS AND FUTURE WORK</b>	<b>54</b>
	<b>ABBREVIATIONS - ACRONYMS</b>	<b>55</b>
	<b>REFERENCES</b>	<b>58</b>

## LIST OF FIGURES

2.1	Transformers' Architecture . . . . .	17
2.2	BERT is a trained Transformer Encoder stack . . . . .	18
2.3	BERT only uses the encoder part of the original Transformer network . . . . .	19
2.4	BERT's Input . . . . .	20
2.5	Predicting the word in a sequence . . . . .	21
2.6	Next Sentence Prediction . . . . .	22
3.1	Raptarchis Law's Structure . . . . .	23
3.2	Nomothesi@ Example . . . . .	25
3.3	Nomothesi@ Document Example from 1990 . . . . .	25
3.4	EuroParl Example . . . . .	26
3.5	EUR-LEX Dataset Example . . . . .	27
3.6	EU Framework Directive . . . . .	27
3.7	EUROVOC Descriptor "1085" . . . . .	28
3.8	EUROVOC Descriptor Example 1 . . . . .	29
3.9	EUROVOC Descriptor Example 2 . . . . .	29
3.10	Plenary Session Body . . . . .	30
3.11	Plenary Session Information . . . . .	31
3.12	Google Cloud . . . . .	33
3.13	Google Compute Engine . . . . .	34
3.14	TFRecord Format . . . . .	35
3.15	Google Cloud Storage . . . . .	35
3.16	TPU v3-8 at a glance . . . . .	36
3.17	Tensorflow and Anaconda . . . . .	36
3.18	Pretraining Script . . . . .	37
3.19	Pretraining Snapshot . . . . .	38
3.20	Saved files . . . . .	38
3.21	Hugging Face . . . . .	39
3.22	Model Card . . . . .	39
4.1	Example from Annotated File . . . . .	41
4.2	Example from CoNLL file . . . . .	42
4.3	GreekLegalBERT v2 Tokenizer Example . . . . .	46
4.4	GreekLegalBERT v2 Indexes Example . . . . .	46
4.5	Sentence Pair Classification . . . . .	47
4.6	BERT Inputs Example . . . . .	47
4.7	True Positives-Negatives, False Positives-Negatives . . . . .	51

## LIST OF TABLES

3.1	Raptarchis Sets Statistics . . . . .	24
3.2	Raptarchis Categories Statistics . . . . .	24
3.3	Information about EuroParl Dataset . . . . .	26
3.4	EUR-LEX Sets Statistics . . . . .	30
3.5	Capacities of all preprocessed datasets . . . . .	31
4.1	NER Dataset Statistics . . . . .	43
4.2	NER Fine Tuning Hyperparameters . . . . .	44
4.3	NER Fine Tuning Best Hyperparameters . . . . .	44
4.4	NLI Sentence Example . . . . .	45
4.5	NLI Dataset Statistics . . . . .	46
4.6	NLI Fine Tuning Hyperparameters . . . . .	47
4.7	NLI Fine Tuning Best Hyperparameters . . . . .	48
4.8	Capacities of remaining Raptarchis . . . . .	48
4.9	Capacities of new fine tuning Raptarchis dataset . . . . .	48
4.10	Difference of Labels between Raptarchis datasets . . . . .	49
4.11	Multiclass Classification on Raptarchis Fine Tuning Hyperparameters . . . . .	49
4.12	NER Results . . . . .	52
4.13	NLI Results . . . . .	53
4.14	Multiclass Classification on Raptarchis Results . . . . .	53

## PREFACE

The present thesis is part of the requirements for the acquisition of a Bachelor's degree in Department of Informatics and Telecommunications of National and Kapodistrian University of Athens. The major purpose of this thesis is to train the BERT model with more legal data than it has previously been trained with, to experiment with specific NLP tasks, and to compare our results to those of other Greek models.

Working on this project validated our early interest in Artificial Intelligence, particularly Natural Language Processing. It motivated us to learn more about neural networks, making this field even more appealing. Without this thesis, deciding on our true passions and choosing a general field to pursue in later life would be much more difficult.

We never anticipated working with such a huge and state-of-the-art model, like BERT, so early in our university studies. This journey was gratifying in every way. We will make every effort to communicate our opinions and findings to you.

# 1. INTRODUCTION

It was three years ago, when the field of Natural Language Processing underwent a crucial paradigm change, after the release of Google's BERT [15], which achieved state-of-the-art results on eleven distinct NLP tasks. BERT used masked language modeling techniques to introduce bidirectional contextual representations learning, which had previously been missing in capturing the contextual representations.

In recent years, the application of NLP in legal domain has witnessed a tremendous increase of interest. Many researchers and tech businesses are developing legal applications with NLP skills using deep learning techniques. Following a well-structured procedure with well-defined practices in an area like law, makes it ideal for NLP applications.

Transformer [28] based language models, such as BERT, use unsupervised pretraining approaches on huge text to learn the features of languages and the benefits of transfer learning to swiftly adapt to downstream NLP tasks by finetuning. On the downstream NLP task, BERT offers the benefits of transfer learning, easier finetuning, and faster computation at the finetuning stage. In the pretraining procedure, BERT uses self-supervised learning and the WordPiece embedding vocabulary, which comprises roughly 30K tokens. This paves the way for the BERT Model to increase its performance on domain-specific NLP tasks. This includes additional pretraining of the BERT model on the legal text by initializing with the pretrained general BERT Model, which is said to boost performance on domain-specific NLP tasks.

Along these lines, the present work focuses on three different tasks: *Named Entity Recognition (NER)*, which falls under the Token Classification tasks, *Natural Language Inference (NLI)*, which falls under the Sequence Classification tasks and *Multiclass Legal Text Classification on Raptarchis*.

Token-level classification entails assigning a label to each token. A part-of-speech tagger, for example, will categorize each word as a distinct part of speech, while *NER* will classify each word as a single entity (like Person, Organization, Location etc). The output will include a label for each token (element in the sequence). Sequence classification is the task of predicting a class label given a sequence of observations. *NLI*, for example, is the task of identifying whether a hypothesis-sentence given a premise-sentence is true (entailment), untrue (contradiction), or undecided (neutral). *Multiclass Text Classification* is the process of assigning a set of predetermined categories to unstructured text, and text classifiers can be used to organize, structure, and categorize almost anything.

All of these techniques have, of course, been used in the past. Nonetheless, compared to more commonly spoken languages, publicly available resources for Greek NLP remain quite scarce. The resources for deep learning in Greek Legal NLP are much more scarce. To our knowledge, this is the first time this work has been attempted to this extent, with a model being trained with such a large volume of legal data and a comparison of how well it performs against these various tasks.

Our documentation is mostly made up of four explanatory chapters, which are described below:

- In Chapter 2, we provide background material about BERT model. We discuss the origins of the term, the model's architecture, how exactly it operates, and we go over the tasks that it has been pre-trained on in detail. We also present and highlight the differences of the other two Greek BERT models with which we will be compared.

- In Chapter 3, we describe the datasets that we utilized to train our model and the processes that we had to take to create the GreekLegalBERT v2 model. Firstly, we will go over the preprocessing steps that all legal corpora must go through before they can be used, as well as the vocabulary and Tensorflow records creation. After that, we continue with the actual training of the model on Google Cloud Platform, including all of the Google Cloud Setup in great detail.
- In Chapter 4, we present the three tasks we experiment with, NER, NLI and Multi-class Classification in Raptarchis, as well as the fine-tuning datasets and details of implementation. We examine our model's performance in Masked Language Modeling and compare the analytic results of our model in these tasks to a general-purpose Greek-BERT model and the initial edition of Greek-Legal-BERT.
- In Chapter 5, we expand on our findings and discuss the possibilities for further work.

## 2. BACKGROUND AND RELATED WORK

### 2.1 BERT Model

#### 2.1.1 What is BERT?

BERT is an open source machine learning framework for Natural Language Processing (NLP). BERT is intended to assist computers in the process of understanding the meaning of ambiguous language in text by establishing context through the use of surrounding text. The BERT framework was pre-trained using text from Wikipedia (2.500 million words) and BooksCorpus [30] (800 million words) and can be fine-tuned with question and answer datasets.

It is aimed to pre-train deep bidirectional representations from unlabeled text by conditioning on both left and right context at the same time. Due to this, the pre-trained BERT model can be fine-tuned with just one additional output layer to provide state-of-the-art models for a variety of NLP tasks, as we accomplished.

##### 2.1.1.1 B: Bi-directional

The majority of models are uni-directional. They can traverse over the word's context window from only left to right or right to left. Only in one direction, but not both at the same time. BERT, on the other hand, is a model that is "deeply bidirectional". It can see the entire sentence on either side of a word contextual language modeling and all of the words almost at once. BERT basically gives us contextual-bidirectional embeddings [8]:

- **Contextual:** A word's embeddings are not static. That means, they depend on the context of the surrounding words. The two embeddings of the word "bird" in a sentence like "One bird was flying below another bird." will be different.
- **Bi-Directional:** This feature allows the model to understand the context of a word by looking at everything around it (left and right of the word). During the training phase, it can learn information from both the left and right sides of a token's context.

This is essential because a word's meaning can frequently alter as a sentence develops [22]. Each word added augments the overall meaning of the word that the NLP algorithm is focusing on. The more words that are present in total in each sentence or phrase, the more ambiguous the word in focus becomes. By reading bidirectionally, BERT adjusts for the augmented meaning by accounting for the effect of all other words in a phrase on the focus word and reducing the left-to-right momentum that biases words towards a given meaning as a sentence progresses.

Let's have a look at an example to demonstrate this. This example contains two sentences, both of which contain the word "bank":

«We went to the *river bank*.»

«I need to go to the *bank* to make a *deposit*.»



We will make an error in at least one of the two examples if we try to guess the nature of the term "bank" by merely looking at the left or right context. One solution is to consider both the left and right contexts before making a prediction. That is precisely what BERT achieves!

### 2.1.1.2 ER: Encoder Representations

The text must be encoded in a format that the model can understand. It maps an input sequence of symbol representations to a sequence of continuous representations. It is composed of a stack with six identical layers. Each layer is divided into two sub-layers. A multi-head self-attention mechanism is the first layer. The second layer is a simple, position-wise fully connected Feed-Forward Network. We employ a residual connection around each of the two sub-layers, followed by Layer Normalization. Layer normalization's main feature is that it normalizes the inputs across all features. To handle a range of tasks, our input representation can clearly represent both a single sentence and a pair of sentences (e.g. Named Entity Recognition, in one token sequence in the form of transformer representations).

### 2.1.1.3 T: Transformers

BERT uses "Transformers" [28] and "Masked Language Modeling". Transformer is made up of two different mechanisms: an encoder that reads the text input and a decoder that produces a prediction for the task. Only the encoder mechanism is required because BERT's purpose is to generate a language model.

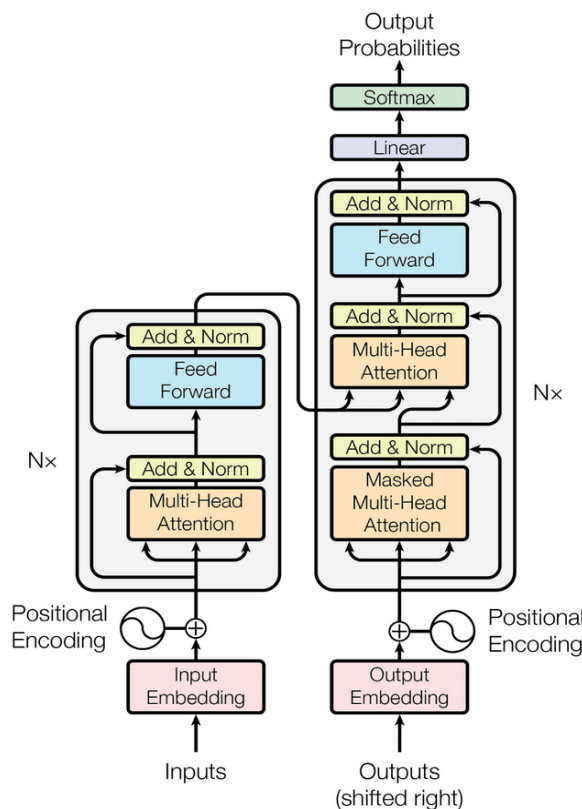


Figure 2.1: Transformers' Architecture

One of the major problems with Natural Language Understanding in the past has been not being able to understand in what context a word is referring to. For instance, pronouns. In a conversation, it's quite simple to lose track of who's talking about what. Even humans find it challenging to keep track of who is being mentioned in a conversation all of the time. Search engines behave similarly, except they have a hard time keeping track of when you say he, they, she, us, it, and so on. As a result, Transformers' concentration is on pronouns and the meanings of all the words that go together to figure out who is being talked to or what is being discussed in any particular context.

The target word cannot perceive itself because of Masked Language Modeling. The mask is required because it stops the under-focus word from seeing itself. BERT just guesses what the missing word is once the mask is on. It's also a part of the fine-tuning process.

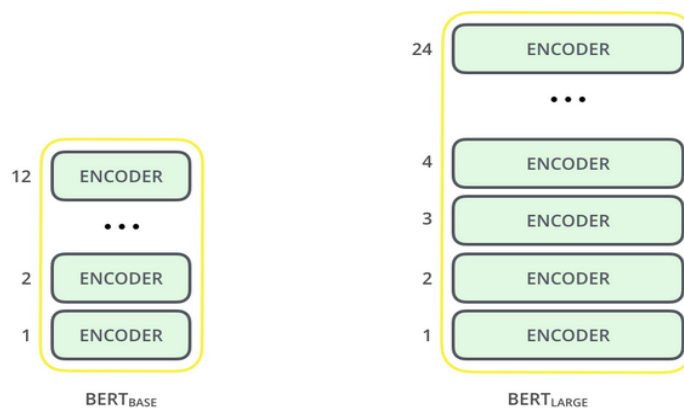
Futhermore, BERT is based on the Transformer model architecture. BERT's model details will be presented very soon, but in general: A Transformer works by performing a small, constant number of steps. In each step, it applies an attention mechanism to understand relationships between all words in a sentence, regardless of their respective position.

### 2.1.2 BERT's Architecture

BERT architecture builds on top of Transformer. It is available in two unique variations:

BERT Base	BERT Large
Layers – 12	Layers – 24
Hidden State – 768	Hidden State – 1024
Self Attention Heads – 12	Self Attention Heads – 16
Total Parameters – 110 Million	Total Parameters – 340 Million

An important note is the fact that BERT-Base was trained on 4 cloud TPUs for 4 days and BERT-Large was trained on 16 TPUs for 4 days. Futhermore, all of these layers are Encoder-only blocks. Here, we should mention that we used the BERT Base model to train our model with more data.



**Figure 2.2: BERT is a trained Transformer Encoder stack**

BERT's architecture is derived from Transformers. There are multiple stacked encoder cells inside a BERT [17]. The encoder cells are used to read the input sentence and the

decoder cells are used to predict the output sentence inside a transformer, in a word by word manner. However, in the case of BERT, since we only need a model that reads the input sentence and generates some features that can be used for various NLP tasks, only the encoder part of the transformer is utilized. As a result, BERT's output is an embedding rather than a textual output.

This already tells us a lot about BERT. Because it uses the encoder, it's not meant for tasks like text generation or translation. Although it can be trained on a variety of languages, it is not a machine translation model in and of itself. Similarly, it can still predict words, so it can be used as a text generating model, but that's not what it's optimized for.

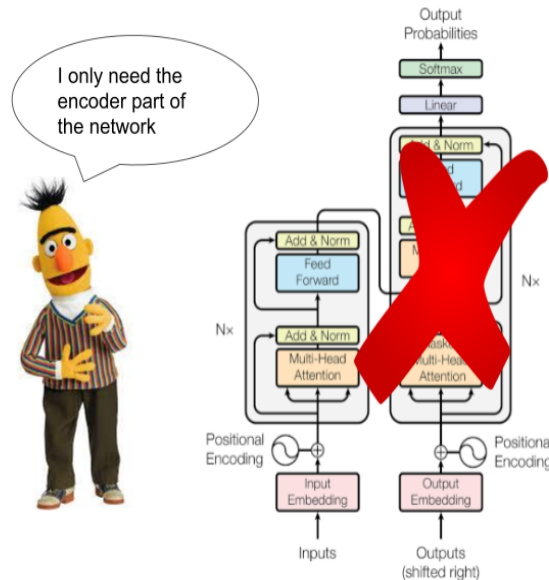


Figure 2.3: BERT only uses the encoder part of the original Transformer network

### 2.1.3 How does it work?

Now that we've learned about BERT's architecture, let's look at how to utilize it to generate output from some input text.

#### 2.1.3.1 Input

Each input embedding is composed of three embeddings [25]:

1. **Token embeddings:** A pair of sentences, known as sequences, and two unique tokens, [CLS] and [SEP], make up the input. So, in this example, for the two sentences "my cat is funny" and "she likes jumping," BERT converts the sequence into tokens first (using Wordpiece Tokenization [18]), then inserts the [CLS] token in the beginning and the [SEP] token at the beginning and end of the second sentence, resulting in the input:

*[ '[CLS]', 'my', 'cat', 'is', 'funny', '[SEP]', 'she', 'likes', 'jumping', '[SEP]' ]*

BERT's Wordpiece Tokenization separates words like "jumping" into "jump" and "##ing." This is beneficial in two ways:

- It helps to reduce the size of our vocabulary since we don't have to retain all of the many forms of terms in our vocabulary, such as **jumping**, **jumps**, and **jumper**.
  - It assists us with words that are not in our vocabulary. If jumps isn't in the vocabulary, for example, we might still have embeddings for **jumb** and **##s**.
2. **Segment embeddings:** These are used to help BERT distinguish between the different sentences in a single input. For words from the same sentence, the members of this embedding vector are all the same, but the value varies if the sentence is different. So, for the previous example, the segment embeddings will be as follows:

**[0, 0, 0, 0, 0, 0, 1, 1, 1, 1]**

Observe how all of the elements corresponding to the word in the first sentence have the same element 0 while all of the elements corresponding to the word in the second sentence have the same element 1.

3. **Positional embeddings:** These are the embeddings that are used to specify the position of the words in the sequence. So we have a constant matrix with a pre-defined pattern. The number of columns in this matrix is 768. The first row of this matrix represents the embedding for the token [CLS], the second row represents the embedding for the word "my," the third row represents the embedding for the word "cat," and so on.

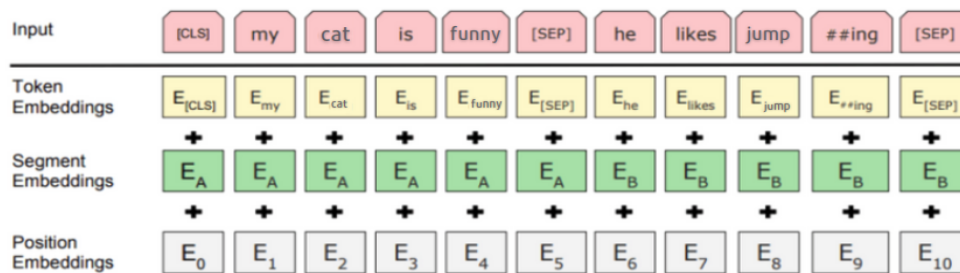


Figure 2.4: BERT's Input

### 2.1.3.2 Output

As previously stated, the input to BERT is typically a sequence of words, and the output is a sequence of vectors. BERT base generates a 768-dimensional output internally. Finally, we take this output and feed it into a feed-forward network such as a Dense layer and a softmax or sigmoid as the activation function. This is something that allows us to perform a variety of tasks based on its output.

### 2.1.4 Tasks that BERT has been pre-trained on

The model was trained on two different types of tasks: *Masked Language Modeling (MLM)* and *Next Sentence Prediction (NSP)*. Let's take a closer look at these training approaches and see what we can learn.

### 2.1.4.1 Masked Language Modeling

Language Modeling is the task of predicting the next word given a sequence of words. Instead of predicting every next token, Masked Language Modeling masked a fraction of input tokens at random and only those masked tokens are predicted [1].



Figure 2.5: Predicting the word in a sequence

Because the masked tokens would never be seen until fine-tuning, the masked words are not always replaced with the masked token [MASK]. As a result, 15% of the tokens are picked at random, and:

- 80% of the time tokens are actually replaced with the token [MASK].
- 10% of the time tokens are replaced with a random token.
- 10% of the time tokens are left unchanged.

### 2.1.4.2 Next Sentence Prediction

Masked Language Models are taught to recognize the connections between words. BERT is also trained on Next Sentence Prediction, which is useful for tasks that require an understanding of the connection between sentences. Given a pair of sentences, it is predicted if the second sentence is the actual next sentence of the first sentence or just a random sentence.

BERT uses a specific [SEP] token to separate sentences, as we saw earlier. The model is fed two input sentences at a time during training, as follows:

- 50% of the time the second sentence comes after the first one.
- 50% of the time it is a random sentence from the full corpus.

BERT must then predict whether the second sentence is random or not, based on the assumption that the random sentence will be unrelated to the first:

```

Input = [CLS] the man went to [MASK] store [SEP]
        he bought a gallon [MASK] milk [SEP]
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
        penguin [MASK] are flight ##less birds [SEP]
Label = NotNext

```

**Figure 2.6: Next Sentence Prediction**

To determine if the second phrase is connected to the first, the entire input sequence is passed through a Transformer-based model, the output of the [CLS] token is transformed into a 2x1 shaped vector using a simple classification layer, and the IsNext-Label is assigned using softmax.

Both Masked Language Modeling and Next Sentence Prediction are used to train the model. This is done in order to reduce the combined loss function of the two techniques; *“better together.”*

## 2.2 Comparison with other BERT Models

We compare our model against two other BERT models that have been pre-trained in Greek and can comprehend them to measure how well or poorly it performs. These models are **“Greek-BERT”** and **“Greek-Legal-BERT-v1”**. The vocabulary in all of these models is uncased. We have to underline that we used the name “GreekLegalBERT v2” for our model, as it is actually an extension of “Greek-Legal-BERT”, since we trained it with more legal data this time.

### 2.2.1 Greek-BERT

Greek-BERT is a model that has the same original BERT architecture, but it also trained on the Greek language. It was trained on 29GB of Greek corpus. Specifically, the Greek part of Wikipedia [4] (0.73GB, 0.08 billion tokens), the Greek part of the European Parliament Proceedings Parallel Corpus [19] (0.38GB, 0.04 billion tokens), and the Greek part of OSCAR [24], a clean version of Common Crawl (27GB, 2.92 billion tokens). It has also a 35.000 tokens vocabulary.

### 2.2.2 Greek-Legal-BERT-v1

Greek-Legal-BERT-v1 is a model following the same original BERT architecture, trained on the Greek legal language. It was trained on 4.5 GB of Greek legal corpus. Specifically, the entire corpus of Greek Legislation is available in Nomothesi@ [10] platform. This corpus consists of numerous laws, announcements and resolutions in the Greek Language. It has a 35.100 tokens vocabulary. Note that of the three models we use, Greek-Legal-BERT-v1 has the smallest pre-training dataset.

### 3. PRETRAIN OF GREEKLEGALBERT V2 MODEL

#### 3.1 Datasets and Preprocessing

The first step in developing an AI system is to figure out what problem it needs to address. The availability of data will have a massive effect on how the system is assembled and which AI techniques are utilized. The *quantity* and *quality* of data available will have an impact on the final product's quality. In this way, one could argue that data availability (if data exists) and accessibility (whether data is available) are the primary motivators for the development of AI-based services. So, let us now describe and examine our datasets, as well as the preprocessing we used to create a combined and cleaned dataset with an 8.6 GB capacity.

##### 3.1.1 Raptarchis

The Raptarchis dataset, also known as RAPTARCHIS47k [13], consisting of approximately 47 thousand legal resources, is a comprehensive collection of Greek legislation dating from the founding of the Greek state in 1834 through 2015. It contains Laws, Royal and Presidential Decrees, Regulations, and Decisions gathered from the Official Government Gazette, which publishes Greek legislation. This collection is one of few official, publicly accessible sources of classified Greek legislation that can be used for classification purposes.

In addition, the Raptarchis dataset offers 3 hierarchical levels of thematic categorization, which, for the purpose of the experiments, are labelled as:

- **Volume**, it is the first and broader level of thematic categorization. It consists of 47 different classes and is divided into Chapters.
- **Chapter**, it is the second level of thematic categorization. It consists of 389 different classes and is divided into Subjects.
- **Subject**, it is the third, final and more specialized level of the thematic categorization. It consists of 2285 different classes.

title	type	year	law_id	leg_uri	volume	chapter	subject	header	articles
88. ΑΠΟΦΑΣΗ ΥΠΟΥΡΓΩΝ	ΑΠΟΦΑΣΗ	1988	NaN	NaN	ΣΤΡΑΤΟ Σ ΞΗΡΑΣ	ΑΣΦΑΛΙΣΤΙΚΟΙ ΟΡΓΑΝΙΣΜΟΙ ΣΤΡΑΤΟΥ	ΜΕΤΟΧΙΚΟ ΤΑΜΕΙΟ ΣΤΡΑΤΟΥ	88. ΑΠΟΦΑΣΗ ΥΠΟΥΡΓΩΝ ΠΡΟΕ ΔΡΙΑΣ ΤΗΣ ΚΥΒΕΡΝΗΣΗΣ...	0

Figure 3.1: Raptarchis Law's Structure

As we can observe at Figure 3.1, a law of Raptarchis dataset encapsulates the header column, which contains detailed info about the law, while in the articles column there is a probability, of approximately 65%, of containing articles being associated with the law. Each and every law has one unique value of the following categories: Volume, Chapter, Subject. The columns title, type, year, and possibly law\_id appear at the beginning of each law in the header column. Additionally, the column leg\_uri contains a link that refers to the law (remark that only 3% of the laws have a link).

**Table 3.1: Raptarchis Sets Statistics**

<b>Set</b>	<b>No of Documents</b>	<b>Avg. words</b>
Train	28,536	600
Development	9,511	574
Test	9,516	595

As indicated in Table 3.1, the Raptarchis dataset is divided into three distinct sets: train, development, and, of course, test set. We should point out that during our model's pre-training, we only collected the train set and left the other two out for the time being.

**Table 3.2: Raptarchis Categories Statistics**

<b>Level</b>	<b>Total</b>	<b>Frequent</b>	<b>Few-Shot</b>	<b>Zero-Shot</b>
Volume	47	47	0	0
Chapter	389	333	53	3
Subject	2285	712	1431	142

Furthermore, the Raptarchis dataset contains three categories of legal document resources for each thematic level, based on the frequency of their labels in the training and testing set: There are three types of shots: frequent, few, and zero. Frequent classes appear in training documents more than 10 times. In 1 to 10 training documents, only a few shooting courses appear. The development and/or test sets have zero shot classes, but the training set hasn't. In the above Table 3.2, we can inspect the distribution of the three before mentioned categories into Volume, Chapter and Subject level.

### 3.1.2 Nomothesi@ Platform

There has been a growing interest in making government data open and accessible to the public during the last decade. More recently, breakthroughs in the field of NLP have enabled the successful processing of textual data through the application of deep learning. Legislation-related data is an essential type of government data. Legislation affects every area of people's lives and is always evolving, resulting in a vast network of interconnected legal documents. As a result, it is critical for a government to provide services that make legislation easily accessible to the general people, with the goal of informing citizens, enabling them to defend their rights, or allowing them to use legislation as part of their employment.

Following in the footsteps of other European countries, the Nomothesi@ project intends to modernize the way Greek legislation is made public while also offering high-quality legal services. There is currently no other endeavor in Greece, nor any relevant decisions made by government institutions or administration, that considers this perspective on legislation.

Nomothesi@, a platform that makes Greek legislation available on the Web as linked open data, was built on the basis of the aforementioned principles. Nomothesi@ gives open access to over 12.000 pieces of legislation in a library with over 124.400 connections and over 8.000 unique entities, such as geospatial entities, persons, organizations, and geographic landmarks (points of interest). Therefore, we used the entire corpus of Greek Legislation in Nomothesi@ platform, which includes various laws, announcements and resolutions in Greek Language. An example of this dataset could be observed right below:



Προβολή ΦΕΚ  
 ΠΡΟΕΔΡΙΚΟ ΔΙΑΤΑΓΜΑ ΥΠ' ΑΡΙΘΜ. 66  
 Οργανισμός Σχολής Μονίμων Υπαξιωματικών Ναυτικού.  
 Ο ΠΡΟΕΔΡΟΣ  
 ΤΗΣ ΕΛΛΗΝΙΚΗΣ ΔΗΜΟΚΡΑΤΙΑΣ  
 Έχοντας υπόψη:  
 α. Τα άρθρα 1 παρ. 1 και 2 του α.ν. 428/1937 «Περί της εν τω Β. Ναυτικού Εκπαίδευσεως» (Α' 15),  
 β. Την υπ' αριθμ. Υ24/6-10-2015 απόφαση του Πρωθυπουργού «Ανάθεση αρμοδιοτήτων στον Αναπληρωτή Υπουργό  
 Εθνικής Άμυνας Δημήτριο Βίτσα» (Β' 2144).  
 γ. Την υπ' αριθμ. 45/4-5-2015 απόφαση του Ανώτατου Ναυτικού Συμβουλίου.  
 δ. Την από 5-6-2015 εισήγηση του Αρχηγού του Γενικού Επιτελείου Ναυτικού.  
 ε. Την από 18-6-2015 σύμφωνη γνώμη του Αρχηγού του Γενικού Επιτελείου Εθνικής Άμυνας.  
 στ. Την υπ' αριθμ. 93/2015 εισήγηση του Προϊσταμένου Οικονομικών Υπηρεσιών του ΥΠΕΘΑ.  
 ζ. Το γεγονός ότι από τις διατάξεις του παρόντος δεν προκαλείται επιπλέον δαπάνη σε βάρος του Κρατικού  
 Προϋπολογισμού.  
 η. Την υπ' αριθμ. 65/2017 γνωμοδότηση του Ε' Τμήματος Συμβουλίου της Επικρατείας, μετά από πρόταση του  
 Αναπληρωτή Υπουργού Εθνικής Άμυνας.  
 Με πρόταση του Αναπληρωτή Υπουργού Εθνικής Άμυνας, αποφασίζουμε:  
 ΚΕΦΑΛΑΙΟ Α  
 ΓΕΝΙΚΕΣ ΔΙΑΤΑΞΕΙΣ  
 Άρθρο 1  
 Νομική Φύση της Σχολής  
 Η Σχολή Μονίμων Υπαξιωματικών Ναυτικού (ΣΜΥΝ) είναι Ανώτερη Στρατιωτική Σχολή Υπαξιωματικών (ΑΣΣΥ), ανήκει  
 στην ανώτερη βαθμίδα της τριτοβάθμιας εκπαίδευσης, οργανώνεται δε και λειτουργεί σύμφωνα με τις ισχύουσες  
 διατάξεις και τον παρόντα Οργανισμό (Ο/ΣΜΥΝ).  
 Άρθρο 2  
 Σκοπός Λειτουργίας  
 Σκοπός της λειτουργίας της ΣΜΥΝ είναι να παρέχει στο Πολεμικό Ναυτικό (ΠΝ) Υπαξιωματικούς με την κατάλληλη  
 αγωγή και εκπαίδευση, που θα είναι ικανοί να ανταποκριθούν στα καθήκοντα της ειδικότητας και της εξειδίκευσης  
 τους, για την κάλυψη των αναγκών του.

Figure 3.2: Nomothesi@ Example

In general, as you can see, the dataset is clean to a pretty decent level, and no special preprocessing or character removal is required. However, we observed that the more you go to previous years, the more the purity of the documents is affected. For example, the image above is from 2017, while the photo below is since 1990:

ΕΦΗΜΕΡΙΣ ΤΗΣ ΚΥΒΕΡΝΗΣΕΩΣ  
 ΤΗΣ ΕΛΛΗΝΙΚΗΣ ΔΗΜΟΚΡΑΤΙΑΣ  
 Α Θ Η Ν Α  
 & ΙΑΝΟΥΑΡΙΟΥ 1990  
 ΑΡΙΘΜΟΣ ΦΥΛΛΟΥ  
 ΤΕΥΧΟΣ ΠΡΩΤΟ 3  
 ΠΕΡΙΕΧΟΜΕΝΑ  
 ΠΡΟΕΔΡΙΚΑ ΔΙΑΤΑΓΜΑΤΑ  
 1. Προστασία των δικαιωμάτων των εργαζομένων σε περί-  
 πτωση αφερεγγυότητας του εργοδότη. .... ε 1  
 2. Αναγνώριση και κατάσταση του Σεβασμιωτάτου Μητρο-  
 πολίτη Λαρίσης και Τυρνάβου Δημητρίου Μπεκιάρη. .. 2  
 3. Σύσταση και ακατάργηση Γενικών [Γραμματειών στο  
 Υπουργείο Μεταφορών και Επικοινωνιών. 3  
 4. Χορήγηση οικονομικής ενίσχυσης σε ορισμένες κατηγορίες  
 ναυτικών λόγω εορτών ΧΡΙΣΤΟΥΓΕΝΝΩΝ 1989. ... 4  
 ΠΡΑΞΕΙΣ ΥΠΟΥΡΓΙΚΟΥ ΣΥΜΒΟΥΛΙΟΥ  
 Αύξηση των θέσεων των Ειδικών Συμβούλων - Ειδικών Συ-  
 νεργατών και των μετακλητών διοικητικών υπαλλήλων  
 - στο Πολιτικό Γραφείο του Υπουργού Μεταφορών και Επι-  
 χεινωμένων.  
 Αύξηση των θέσεων των Ειδικών Συμβούλων -- Ειδικών Συ-  
 νεργατών, των μετακλητών διοικητικών και αποσπασμέ-  
 νων υπαλλήλων στο Πολιτικό Γραφείο του Υπουργού Ερ-  
 γασίας. ... α α α ε α ε α α σ 6  
 Αύξηση των θέσεων των Ειδικών Συμβούλων - Ειδικών Συ-  
 νεργατών και των μετακλητών και αποσπασμένων διοικη-  
 τικών υπαλλήλων στο Πολιτικό Γραφείο του Υπουργού  
 Περιβάλλοντος, Χωροταξίας και Δημοσίων Έργων. ... 7  
 Αύξηση των θέσεων των Ειδικών Συμβούλων - Ειδικών Συ-  
 νεργατών και των μετακλητών διοικητικών υπαλλήλων  
 στο Πολιτικό Γραφείο του Αναπληρωτή Υπουργού Μετα-  
 φορών και Επικοινωνιών.

Figure 3.3: Nomothesi@ Document Example from 1990

Nevertheless, this is not a general rule, as this phenomenon is not observed in many documents and we considered that if we carry out a special treatment it would affect the performance of the whole dataset.

### 3.1.3 EuroParl

Philipp Koehn's [19] team in Edinburgh was able to collect corpus parallel text from the European Parliament sessions in 11 languages from European Union (Danish, Dutch, English, Finnish, French, German, Greek, Italian, Portuguese, Spanish, and Swedish). Their collection is based on the European Parliament's proceedings, which date back to 1996. Their goal was to use the corpus to create over 100 machine translation systems for any language combination that could be imagined. We took advantage of the chance to expand our Greek corpus because searching Greek raw text with a certain domain-orientation, such as Legal Documents, is a challenging task. Despite the fact that Koehn and his team provide datasets in a variety of languages, we are mainly interested in the Greek part of the European Parliament Proceedings Parallel Corpus, which contains mostly dialogues in Greek about legal and ethical issues.

**Table 3.3: Information about EuroParl Dataset**

Language	Days	Chapters	Speaker Turns	Sentences	Words
Greek (el)	398	3.712	66.928	746.834	27.772.533

The data that makes up the corpus was extracted from the website of the European Parliament and then prepared for linguistic research. After sentence splitting and tokenization the sentences were aligned across languages. As shown in the above Table 3.3, the Greek data, which are merged Parliamentary reports, were recorded for more than a year and finally end up reaching approximately 30 million words.

Κύριε Πρόεδρε, κύριε Επίτροπε, η προκειμένη έκθεση του συναδέλφου Berend ακολουθεί κατά βήμα τη στρατηγική που καθόρισε η Ευρωπαϊκή Επιτροπή, θέτοντας απόλυτα στο προσκήνιο το ζήτημα της αύξησης της ανταγωνιστικότητας. Η διατομεακή στοχοθεσία των διαρθρωτικών ταμείων, όπως φέρ' ειπείν η ενισχυμένη δημιουργία απασχόλησης, η μεγαλύτερη εξασφάλιση της ισοτήτας ευκαιριών και η αύξηση της αειφορίας της απασχόλησης και της ανάπτυξης αναφέρονται μόνον παρεμπιπτόντως. Ο τρόπος αυτός θεώρησης μου φαίνεται αδικαιολόγητος, και θα παρακαλούσα επίσης όπως τα εν λόγω θέματα εξετασθούν πολύ αποτελεσματικότερα στην έβδομη περιοδική έκθεση. Αυτό δεν σημαίνει πως δεν αναγνωρίζω την ανάγκη της δυναμικής του ανταγωνισμού, πόσο μάλλον αφού κι εγώ ο ίδιος τυγχάνω επιχειρηματίας σε μια περιφέρεια του στόχου 1, και μάλιστα στο Βρανδεμβούργο της Ομοσπονδιακής Δημοκρατίας της Γερμανίας, και γνωρίζω πολύ καλά τις ανησυχίες και τις ανάγκες των μικρών και μεσίων επιχειρήσεων. Στις περιφέρειες του στόχου 1 χρειάζεται οπωσδήποτε να θεσπισθούν παράλληλα με αυτό και μέτρα ορισμένου χρόνου, ήτοι μέτρα για τη δημιουργία θέσεων εργασίας, ειδικά προγράμματα για την προαγωγή της βιοποριστικής δραστηριότητας των γυναικών και πρωτοβουλίες για την πρόωθηση της έναρξης της μη μισθωτής δραστηριότητας. Τα μέτρα αυτά θα υποστηριχθούν με κατάλληλες δραστηριότητες από τα διαρθρωτικά ταμεία της Ευρωπαϊκής Ένωσης. Η ενίσχυση της ανταγωνιστικότητας των επιχειρήσεων από μόνη της δεν μπορεί σε καμία περίπτωση να αναπληρώσει την επιζητούμενη συνοχή μεταξύ οικονομικής και κοινωνικής ανάπτυξης, επειδή λείπουν απλώς τα μέσα για μια ανεξάρτητη ανάκαμψη στις εν λόγω περιφέρειες του στόχου 1. Και η εμπειρία ότι η οικονομική ανάπτυξη δεν συμβάλλει από μόνη της στη μείωση της ανεργίας θεμελιώνει, βέβαια, το γεγονός ότι για να εμψυχώσουμε επιπλέον θέσεις εργασίας χρειαζόμαστε μια αύξηση τουλάχιστον κατά 3% του ΑΕΠ.

**Figure 3.4: EuroParl Example**

A Parliamentary report that is submitted to a vote is normally preceded by a debate in which the Commission, representatives from political organizations, and Members of Parliament all have a say. Speaking time is typically limited due to the large number of Members who have requested to speak. A short piece of a random Greek Parliamentary report can be seen in the Figure 3.4 above.

#### 3.1.4 EUR-LEX

EUR-LEX [2] provides online access to European Union (EU) legal documents that is both official and comprehensive. It is updated daily and is available in all of the EU's 24 official languages. We downloaded the EURLEX57K Dataset, containing 57 thousand Greek EU legislative documents from the EUR-LEX portal, tagged with approximately 4.3 thousand labels, or broadly known as concepts, from the European Vocabulary [3] (EUROVOC), which is formatted as follows:

	celex_id	uri	type	concepts	text
0	32001D0008	http://publications.europa.eu/resource/cellar/...	Decision	[1445, 1755, 1857, 192, 5034]	\nΑπόφαση της Επιτροπής\ntης 29ης Δεκεμβρίου 2...
1	32004D0542	http://publications.europa.eu/resource/cellar/...	Decision	[1309, 1598, 1644, 2300, 2771, 4689, 5369]	\nΑΠΟΦΑΣΗ ΤΗΣ ΕΠΙΤΡΟΠΗΣ\ntης 25ης Ιουλίου 2004...
2	32008R1281	http://publications.europa.eu/resource/cellar/...	Regulation	[161, 1644, 1788, 4692, 5367, 888]	\nΚΑΝΟΝΙΣΜΟΣ (ΕΚ) αριθ. 1281/2008 ΤΗΣ ΕΠΙΤΡΟΠΗ...
3	31988R0314	http://publications.europa.eu/resource/cellar/...	Regulation	[1264, 164, 4291]	\n****\nΚΑΝΟΝΙΣΜΟΣ (ΕΟΚ) αριθ. 314/88 ΤΗΣ ΕΠΙ...
4	31988R4095	http://publications.europa.eu/resource/cellar/...	Regulation	[1603, 1605, 1609, 161, 4172]	\nΚΑΝΟΝΙΣΜΟΣ ( ΕΟΚ ) αριθ . 4095/88 ΤΟΥ ΣΥΜΒΟΥ...

Figure 3.5: EUR-LEX Dataset Example

Let's take a look at the different columns one by one:

- **celex-id:**

A CELEX number is a unique identifier for a document. A CELEX number has different parts, which vary slightly depending on the type of document. The most likely scenario is to have the four parts listed below: Sector, Year, Document type, Document number.

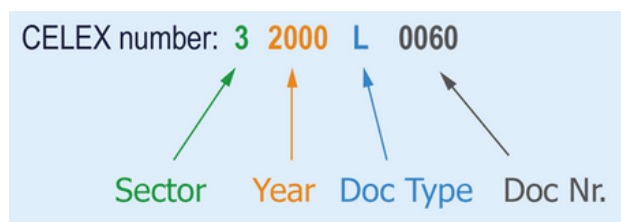


Figure 3.6: EU Framework Directive

– **Sector:** EUR-LEX documents are divided into 12 categories:

1	Treaties
2	International agreements
3	Legal acts
4	Complementary legislation
5	Preparatory documents
6	EU case-law
7	National transposition
8	References to national case-law concerning EU law
9	Parliamentary questions
0	Consolidated texts
C	Other documents published in the Official Journal C series
E	EFTA documents

However, in our dataset, the 57.000 documents start with the number 3 for identifier, indicating that we are only dealing with Legal Acts.

- **Year:** Year is frequently used to indicate when a document was adopted.
- **Document Type:** A descriptor is assigned to each document type. Given the fact that we only have documents with Legal Acts, the following are the types of documents we will encounter:

<b>L</b>	Directives
<b>R</b>	Regulations
<b>D</b>	Decisions

- **Document Number:** Most CELEX numbers end in 4 digits. These reflect different types of information, such as:

**Official number** - e.g. 32017R2394: a sequential number showing when the document was published in the Official Journal. It is assigned by the Publications Office.

**Internal number** - e.g. 52018PC0033: assigned by the author of the document, here the Commission (PC).

**Date of publication** - e.g. 32012A0424(01): the month and day of publication in the Official Journal. It is followed by a sequential number in brackets, as other similar CELEX numbers might have been published on the same day.

- **url:**

URL column corresponds to the unique URL of each individual document. For this reason, we opted not to include it in the final txt file because it adds no useful information.

- **type:**

Because our dataset only contains Legal Acts, the file type can take one of three values: Decision, Regulation, and Directive.

- **concepts:**

We may deduce from a quick search that the numbers in the concepts column correspond to EUROVOC descriptions. EUROVOC is a multilingual, multidisciplinary thesaurus that covers the EU's activities. It includes terms in 23 EU languages (Bulgarian, Croatian, Czech, Danish, Dutch, English, Estonian, Finnish, French, German, Greek, Hungarian, Italian, Latvian, Lithuanian, Maltese, Polish, Portuguese, Romanian, Slovak, Slovenian, Spanish, and Swedish), as well as three candidate EU accession: (македонски (mk), shqip (sq), and српски (sr)). It provides keywords arranged into 21 domains and 127 sub-domains for describing the content of EUR-LEX documents. It can also be used to sort documents contained in EUR-LEX.

Let's take a look at an example from one of our dataset's documents. The list of its concepts is the following: ['1085', '155', '5541', '775', '889', '985'].

Descriptor "1085" corresponds to keyword "Γαλλία".

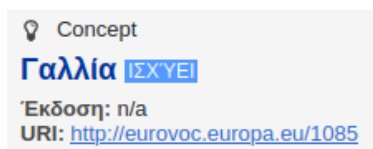


Figure 3.7: EUROVOC Descriptor "1085"

If we examine the document more closely, we will find that the word "Γαλλία" appears multiple times, as well as cognate words such as "γαλλική", "γαλλικών", and so on.

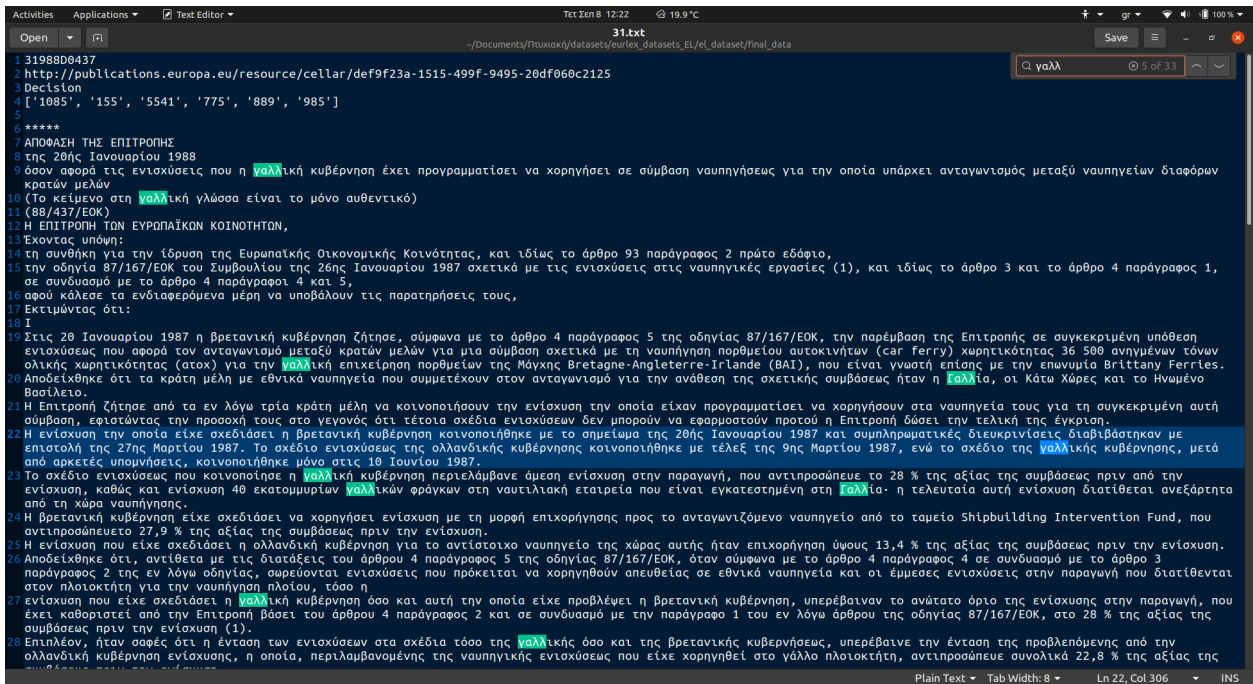


Figure 3.8: EUROVOC Descriptor Example 1

Descriptor "155" corresponds to keyword "ναυπηγικές κατασκευές":



Figure 3.9: EUROVOC Descriptor Example 2

However, we do not know if replacing the descriptor with the keyword would help the model learn better. As above, adding the term "ναυπηγικές κατασκευές" once again, while it has been mentioned approximately forty times in the text, we do not think that it will be provide any further important information. Undoubtedly, if it is not performed frequently enough (e.g. once to three times), another repetition may be beneficial. For this reason, we opted to maintain this information, because in cases that a word is not repeated often it can help it, while in cases that it is repeated quite



often it would not mind another reference. Despite the fact that EUROVOC contains over 7 thousand concepts, the majority of them are rarely used in practice.

- **text:**

This column contains the body of each EUR-LEX document's text.

**Table 3.4: EUR-LEX Sets Statistics**

Sets	No of Documents	Size
<b>Train</b>	45.000	441.5 MB
<b>Development</b>	6.000	53.6 MB
<b>Test</b>	6.000	54.6 MB

As indicated in Table 3.4, the EUR-LEX dataset is divided into three distinct sets: train, development, and, of course, test set. We should point out that during our model's pretraining, we only collected the train set and left the other two out for the time being.

### 3.1.5 Hellenic Parliament Sessions

In this section, we tried to collect all the available minutes of the plenary sessions of the Greek or Hellenic Parliament, from 3 July 1989 to 24 August 2021, including all types and extent of speeches. In all these years the format of files was changing uninterruptedly, from .docx to .doc format or from .txt file with different encodings. We achieve to grab each every session with a custom web-crawler and store them all as text files.

Επίσης, οι υποψήφιοι που ανήκουν στην προαναφερόμενη κατηγορία λαμβάνουν προσαυξήσεις στη μοριοδότησή τους, ανάλογες με το χρονικό διάστημα απασχόλησής τους, ενώ για τις περιπτώσεις που υποψήφιος για το ίδιο χρονικό διάστημα εργάστηκε ως πυροσβέστης εποχικής απασχόλησης και ταυτόχρονα παρείχε εθελοντική υπηρεσία, ως εθελοντής πυροσβέστης, λαμβάνει τις προβλεπόμενες προσαυξήσεις και για τις δύο ιδιότητες. Επισημαίνεται ότι η κάλυψη των προκηρυσσομένων θέσεων αυτής της κατηγορίας γίνεται με βάση τη βαθμολογική σειρά κατάταξής τους αδιακρίτως ιδιότητας. Οι ευνοϊκότερες ρυθμίσεις για την ανωτέρω κατηγορία υποψηφίων συμβάλλουν στη δημιουργία ενός ικανού πλασιού αποτελεσματικής αξιοποίησης υποψηφίων που διαθέτουν πολύτιμη πυροσβεστική προϋπηρεσία και εμπειρία, παρέχοντας ταυτόχρονα τα αναγκαία κίνητρα για ενίσχυση του εθελοντισμού στη χώρα μας, αλλά και την απαραίτητη προσέλευση εποχικού προσωπικού.

Επισημαίνεται ότι για το εργασιακό καθεστώς του εποχικού πυροσβεστικού προσωπικού, : ενημερώθηκε η Εθνική Αντιπροσωπεία την 28-9-2006, κατά τη συζήτηση της 57/93/1-9-2006 επίκαιρης επερώτησης, που κατέθεσαν Βουλευτές του ΠΑ.ΣΟ.Κ., μεταξύ των οποίων και ο ανωτέρω κ. Βουλευτής, καθώς και την 26 και 28-9-2006 κατά τις συνεδριάσεις της Διαρκούς Επιτροπής Δημόσιας Διοίκησης, Δημόσιας Τάξης και Δικαιοσύνης για την επεξεργασία και εξέταση του σχεδίου νόμου του Υπουργείου μας «Αναδιοργάνωση του Πυροσβεστικού Σώματος, αναβάθμιση της αποστολής του και άλλες διατάξεις». Ακόμη, για το θέμα αυτό, ενημερώσαμε την Εθνική Αντιπροσωπεία και κατά τη συζήτηση και ψήφιση επί της αρχής και των άρθρων του αναφερόμενου σχεδίου νόμου, την 31-10 και 1-11-2006 και ανακοινώσαμε στο Σώμα, καθώς και σε εκπροσώπους των εποχικών πυροσβεστικών υπαλλήλων, με τους οποίους συναντηθήκαμε, τις θέσεις μας για την περαιτέρω βελτίωση του εργασιακού θεσμικού πλασιού αυτών. Επίσης, ενημερώθηκε η Εθνική Αντιπροσωπεία την 17-11-2006 και κατά τη συζήτηση της 8/12/ 22-9-2006 επίκαιρης επερώτησης που κατέθεσαν Βουλευτές του ΠΑ.ΣΟ.Κ.

**Figure 3.10: Plenary Session Body**

The Greek Parliament's sessions [5] are accessible to the public; they are typically held on a daily basis and the entire minutes of the plenary debates are published. So during a simple plenary session, members of Parliament make a stand, converse and often the Speaker of the Parliament with its own initiative interrupts in order to give the floor wherever he believes. All this chattering between members of Parliament is documented, as displayed in Figure 3.10, by an official team with excellent ability of stenography.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ  
 ΠΕΡΙΟΔΟΣ ΙΑ'  
 ΣΥΝΟΔΟΣ Γ'  
 ΣΥΝΕΔΡΙΑΣΗ ΡΚΔ'

Τρίτη 8 Μαΐου 2007

ΘΕΜΑΤΑ

Α. ΕΙΔΙΚΑ ΘΕΜΑΤΑ

1. Επικύρωση Πρακτικών, σελ.
2. Ανακοινώνεται ότι τη συνεδρίαση παρακολουθούν μαθητές από το Σχολείο Δεύτερης Ευκαιρίας Κορυδαλλού, σελ.

Β. ΚΟΙΝΟΒΟΥΛΕΥΤΙΚΟΣ ΕΛΕΓΧΟΣ

1. Κατάθεση αναφορών, σελ.
2. Απαντήσεις Υπουργών σε ερωτήσεις Βουλευτών, σελ.
3. Συζήτηση αναφορών και ερωτήσεων:
  - α) Προς τον Υπουργό Αγροτικής Ανάπτυξης και Τροφίμων, σχετικά με τον τρόπο καταβολής των κοινοτικών επιδοτήσεων, σελ.
  - β) Προς τον Υπουργό Υγείας και Κοινωνικής Αλληλεγγύης, σχετικά με την αποκατάσταση ασφαλισμένου του Δημοσίου από δικαστική απόφαση κατά του οργανισμού Ασφαλισμένων του Δημοσίου (ΟΠΑΔ), σελ.

**Figure 3.11: Plenary Session Information**

We should also highlight that at the start of a conventional plenary session of the Greek Parliament, crucial information is captured, such as the date of the session and the topics that will be discussed, as seen in Figure 3.11.

### 3.1.6 Preprocessing

Altogether our five in total datasets structure a large merged corpora of total capacity 8.6GB, as illustrated in the below table.

**Table 3.5: Capacities of all preprocessed datasets**

Dataset	Files	Capacity
Raptarchis Train Set	28.541	0.22GB
Nomothesi@ Platform	97.928	4.9GB
EuroParl	1	0.39GB
Eur Lex Train Set	45.000	0.41GB
Hellenic Parliament Sessions	5.430	2.7GB
<b>Total</b>	<b>176.900</b>	<b>8.6GB</b>

The upcoming decision we had to make was which method of preprocessing we would apply on our combined dataset. So we did follow a specific strategy described below.

**UTF-8 Format:** Ensure several times that every file in our merged dataset is encoded in UTF-8 format, and that any file with a different encoding is converted properly.

**Tokenization:** Split each sentence into words using the most popular delimiter, which is none other than space, in order to apply to a word by word manner our next functions.

**Normalization:** In respect of the removal of punctuation, we decided to remove **only Greek diacritics**, such as *accents* and *dieresis*. This is due to the fact that the datasets (such as EurLEX, Nomothesia, and Hellenic Parliament Sessions) are fairly clean and free of noise. Besides that, there have been instances where the removal of certain punctuations may harm performance in some tasks. For instance, the name of a law might be "2019/75." The information regarding the law would be lost if the punctuation mark '/' is removed. Another example is a date, such as "20/09/1999" or "09.11.2013," which will be just a series of digits with no meaning.

**Merge Files:** Merge each of our datasets' files so that each file has exactly *100.000 lines*, each of which is made up of *250 words*.

### 3.1.7 Tokenization

Before we proceed any further, it would be appropriate to explain tokenization of a corpus. Tokenization is the process of breaking down a phrase, sentence, paragraph, or one or more text documents into smaller components. A token is the name given to each of these smaller components. These tokens can now be any type of **word**, **subword**, or **character**. Different algorithms perform tokenization in various ways, however the example below will give a more accurate sense of the differences between those three.

Consider the following simple phrase:

*"Let us learn tokenization."*

A **word-based** tokenization algorithm will break the sentence into words.

*["Let", "us", "learn", "tokenization."]*

Initially, the risk of missing words in the training data is a significant issue with word-based tokenization algorithms. The model won't recognize variants of words that weren't part of the data on which it was trained if we use word tokens. So, if the model detected "foot" and "ball" in the training data but "football" in the final text, the model will be unable to detect the word and will treat it with the unknown token. Secondly, punctuation poses a different issue; "let" or "let's" will require individual tokens, which is an inefficient approach. This will demand a huge vocabulary in order to guarantee that we have every possible variation of the word.

A **character-based** tokenization algorithm will break the sentence into characters.

*["L", "e", "t", "u", "s", "l", "e", "a", "r", "n", "t", "o", "k", "e", "n", "i", "z", "a", "t", "i", "o", "n", "."]*

Character-based algorithms interpret each character as a token, and more tokens imply more input computations to process each token, which indicates more compute resources are needed. Instead of 5 word-based tokens, you may need to parse 30 tokens for a 5-word phrase. Moreover, working with characters may lead to erroneous word spellings. Learning with characters is also like learning with no semantics since they have no inherent meaning.

A **subword-based** tokenization algorithm will break the sentence into subwords.

*["Let", "us", "learn", "token", "##ization."]*

Subword-based algorithms deliver the ideal balance of character and word-level hybrid representations, allowing it to handle enormous corpora. To put it another way, such algorithms aiming for a tokenization technique that can deal with an infinite possible vocabulary using a finite set of known words. Plus, that subwords retain the semantic features of the token i.e. information per token.

**WordPiece** [18], Byte-Pair Encoding, Unigram, and SentencePiece are some of the most famous subword-based tokenization strategies. In our scenario, we'll concentrate at the WordPiece algorithm because it's widely utilized in language models like BERT. Its technique is to initialize the vocabulary with individual characters in the language, then iteratively add the most frequent combinations of symbols to the vocabulary. The language model is then re-trained on the new vocabulary, and the process is repeated until the target vocabulary size is reached.



### 3.1.8 Generate Vocabulary

After all of this data preprocessing, we've come to a moment where we have 28 distinct files of raw Greek legal text. However, because this input is incompatible with the BERT model's input, we must go to the following steps.

**Generate Vocabulary:** As previously stated, in order to succeed in our vocabulary generation, we will use the **BERTWordPieceTokenizer** from the library of "tokenizers." This tokenizer will be trained with a single and fairly giant text file containing all of our datasets' corpora, resulting in a vocabulary with a predetermined vocabulary size of 35.000. This vocabulary will be extremely crucial in our pre-training and fine-tuning procedures. It consists of all possible characters found by the tokenizer all across the corpus, as well as some special tokens, subwords, and, of course, full words.

**Encode Files:** The transformation from words to ids, or numbers, is the last and most vital task in our preprocessing. In this manner, we will first encode each and every sentence with our BERTWordPieceTokenizer from words to ids, and then add the characteristic [CLS] and [SEP] tokens in the beginning and end of the sentence, respectively. As a consequence, we now have 28 unique files containing encoded raw Greek legal text, which is more than comprehensible from the BERT model.

## 3.2 Google Cloud Setup

Now that we've illustrated our datasets, it's necessary to break down our Google Cloud Setup one step at a time.

### 3.2.1 Why Choose Google Cloud Platform?

Google Cloud Platform (GCP) works similarly to the majority of popular public cloud providers. It offers virtual machines and hardware, which are housed in a regional data center. After that, the regions are separated into distinct zones where data is stored. This enables resources to be preserved close to the customer's location. Failures and latency are also avoided. In addition, there are global, regional and zonal resources, which are gonna meet every customer's needs.



Google Cloud

Figure 3.12: Google Cloud

### 3.2.2 TPU Quotas

In order to attract ambitious researchers and students, Google is giving to each Google Account \$300 in Google Cloud credits for free. Those free credits will assist anyone who is interested in learning more about GCP and completing an assessment.

Furthermore, Google, in collaboration with HuggingFace, offers the Tensorflow Research Cloud Program (TFRC). If a student, a researcher, or a Deep Learning enthusiast doesn't have that much money to spare, TFRC is a godsend. TFRC helps people speed their research by providing free Cloud TPU access, notably by providing them with incredibly powerful hardware for a month at no cost.

### 3.2.3 Create Google Compute Engine (GCE)

The Virtual Machine (VM) idea was established to enable portability within the data center; cloud services like Google Compute Engine (GCE) employ the same format, add a self-provisioning deployment mechanism, and charge users for the resources these VMs consume. Thus, our VM instance will decrease the gap between us and our TPU Node.

Every VM family has predefined machine designs with a certain vCPU to RAM ratio to suit a wide range of workload requirements. We can get away with a low-spec VM instance because our training will be performed on the TPU and does not require a compute or memory optimized unit. So, for our purposes, we built a low-spec VM instance with Debian GNU/Linux 10 that runs on the N2 series, which is part of the 'General Purpose' machine family. Our VM also has two virtual CPUs and eight gigabytes of RAM, and the disk size of ten gigabytes was more than enough. Our VM's setup was, inferentially, one of the cheapest that anyone could construct.



Figure 3.13: Google Compute Engine

### 3.2.4 Create Tensorflow Records (TFRecords)

Because we are working with huge datasets, storing our data in a binary file format can have a great influence on the performance of our import pipeline and, as a result, our model's training time. Binary data takes up less disk space, takes less time to copy, and can be read from disk considerably faster. However, the TFRecord [16] file format's efficiency isn't its only asset. It is optimized for usage with Tensorflow in a range of ways, including making it simple to combine various datasets and integrating effortlessly with the library's data import and preprocessing functionality.

```

INFO:tensorflow:*** Reading from input files ***
I1023 00:11:35.925996 140368328802688 create_pretraining_data.py:448] *** Reading from input files ***
INFO:tensorflow: ./Corpus/files/file46.txt
I1023 00:11:35.926207 140368328802688 create_pretraining_data.py:450] ./Corpus/files/file46.txt
INFO:tensorflow:*** Writing to output files ***
I1023 00:20:43.433646 140368328802688 create_pretraining_data.py:459] *** Writing to output files ***
INFO:tensorflow: ./TFRecords/file46.tfrecord
I1023 00:20:43.441756 140368328802688 create_pretraining_data.py:461] ./TFRecords/file46.tfrecord
INFO:tensorflow:*** Example ***
I1023 00:20:43.475451 140368328802688 create_pretraining_data.py:151] *** Example ***
INFO:tensorflow:tokens: [CLS] αρχηγος της νεας δημοκρατίας και λέει « κοκκινή γραμμή για εμάς είναι το βου ##κου ##ρε ##στι » , ασφαλώς και
αποδεχεται ινες συνθετη ονομασια , ασφαλώς και αποδεχεται την εκχώρηση του ονοματος [MASK] μακεδονια » στα σκοπια . οταν , ομως , λενε να
απαλει ##φθουν [MASK] [MASK] [MASK] [MASK] αναφορες στο συνταγμα των σκοπιων , αυτα είναι γελοιο ##τητες . διοτι οι σκοπια ##νοι [MASK]
σημερα να απαλει ##ψουν τις αλυ ##τρω ##τικες αναφορες στο συνταγμα τους [MASK] η ελλαδα να κανει το εθνικο εγκλημα της αρσης της αρνη
##σικ ##υ ##ριας [MASK] να μην υπαρχει [MASK] βετο και αφου τα σκοπια μπουν στο νατο , ξανα ##ψηφ ##ιζουν καινουργιο [MASK] και λενε οτι
διεκδικουν τη θεσσαλονικη βελτιωνουν ( [MASK] σημειο αυτο κτυπαι το κουδουνι ληξεως του [MASK] ομιλιας του κυριου ##νουμενων ) η 626 ,
[MASK] , για το θεμα της ονομασιας του κρα ##τι ##διου των σκοπιων είναι μια : ελληνικο [MASK] , να μην [MASK] [MASK] σκοπια [MASK] νατο .

```

**Figure 3.14: TFRecord Format**

The `create-pretraining-data.py` script, which we obtained from BERT's official code, converts data to TFRecord format with ease. To reduce computational waste from padding, the formerly mentioned script will concatenate segments until they reach the maximum sequence length, which was proposed by Devlin as 512. The script ran for a couple of hours for our data conversion, and we ended up with 56 TFRecords with a total capacity of 26 GB from an initial 8.6 GB dataset of 56 files. We can justify tripling this capacity because, as shown in the above figure, a TFRecord holds a plethora of additional information such as input ids, input mask, segment ids, next sentence labels, and, of course, the positions, ids, and weights of masked language modeling.

### 3.2.5 Create Google Cloud Storage (GCS)

We had to choose one of the following storage types for our bucket when constructing it in GCS: Standard -which can be either Regional or Multi-Regional-, Nearline, or Coldline. The most common approach, that we also opted, is Standard, which allowed us to store our bucket in a single Google Cloud Region or across multiple Regions. Because it provides high-performance and high-availability storage, this is valuable in a variety of applications.



**Figure 3.15: Google Cloud Storage**

After this procedure we established the directories `data/` and `model/` in our bucket. It took us a week to upload our 56 Tensorflow Records (the entire model's input dataset) to the first folder, which had a total capacity of 26GB. The second folder contains our model's vocabulary as well as the configuration file, which is essential for our model's training. Note that during training, our model's checkpoints will be stored every 20.000 steps, or roughly every 3 hours, in that exact folder.

### 3.2.6 Create Tensor Processing Unit (TPU) Node

TFRC kindly invited us to utilize different TPUs for our model’s pretraining procedure in our situation. In order to achieve the greatest potential performance, we chose the most powerful TPU among our options. We use a single on-demand TPU v3-8 for pretraining of our GreekLegalBERT v2 model. To avoid latency issues and reduce Google’s Service costs, we set our TPU in the *europa west4-a* area, which means it was located in Netherlands.

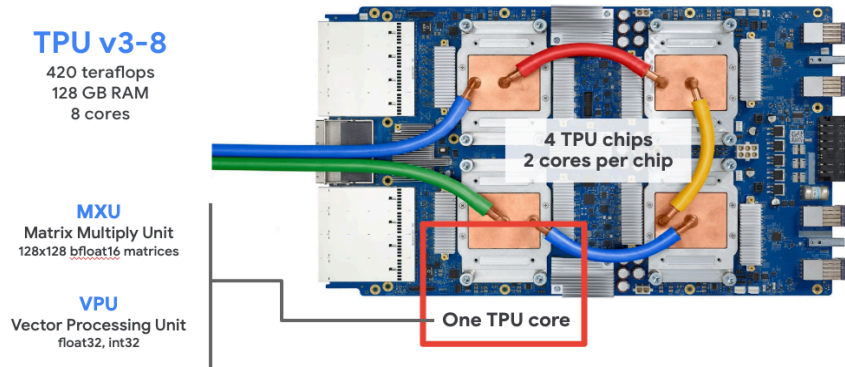


Figure 3.16: TPU v3-8 at a glance

The TPU v3-8 type defines a TPU node with 8 TPU v3 cores and 128 GiB of total TPU memory. In terms of performance, it can handle 420 teraflops, which is quite outstanding.

### 3.2.7 Prepare Virtual Machine (VM) Environment

In order to train our model, we had to first set up some basic libraries. So we installed the entire version of **Anaconda**, which is a very useful program that can literally save a lot of time by combining all of the typical packages used in scientific computing into a single package without having to worry about loading them all separately with their dependencies. **Python 2.7.1** was installed via Anaconda, and we also installed **TensorFlow GPU 1.15** and **CUDA 10.0**. Finally, we cloned official **BERT’s code** from Github in order to execute our python script, which will train our model.



Figure 3.17: Tensorflow and Anaconda

### 3.3 Model Pretraining

We were more than prepared to begin our model pretraining procedure once we had verified that everything in our GCP worked like clockwork.

#### 3.3.1 Pretraining Parameters

```

tpu_cluster_resolver = TPUClusterResolver(tpu='greek-legal-bert-v2-tpu', zone='europe-west4-a')

# SETUP FOLDERS
with tf.Session(tpu_cluster_resolver.get_master()) as session:
    print(tpu_cluster_resolver.get_master())
    HOME_PATH = "gs://greek_legal_bert_v2" # @param {type:"string"}
    MODEL_DIR = "model" # @param {type:"string"}
    PRETRAINING_DIR = "data" # @param {type:"string"}
    VOC_FNAME = "bert-vocab.txt" # @param {type:"string"}

# Input data pipeline config
TRAIN_BATCH_SIZE = 256 # @param {type:"integer"}
MAX_PREDICTIONS = 75 # @param {type:"integer"}
MAX_SEQ_LENGTH = 512 # @param {type:"integer"}
MASKED_LM_PROB = 0.15 # @param

# Training procedure config
EVAL_BATCH_SIZE = 256
LEARNING_RATE = 1e-4
TRAIN_STEPS = 1000000 # @param {type:"integer"}
EVAL_STEPS = 10000
SAVE_CHECKPOINTS_STEPS = 20000 # @param {type:"integer"}
NUM_TPU_CORES = 8
BERT_GCS_DIR = "{}/{}".format(HOME_PATH, MODEL_DIR)
DATA_GCS_DIR = "{}/{}".format(HOME_PATH, PRETRAINING_DIR)
VOCAB_FILE = os.path.join(BERT_GCS_DIR, VOC_FNAME)
CONFIG_FILE = os.path.join(BERT_GCS_DIR, "bert_config.json")
INIT_CHECKPOINT = tf.train.latest_checkpoint(BERT_GCS_DIR)
bert_config = modeling.BertConfig.from_json_file(CONFIG_FILE)
input_files = tf.gfile.Glob(os.path.join(DATA_GCS_DIR, '*'))
log.info("Using checkpoint: {}".format(INIT_CHECKPOINT))
log.info("Using {} data shards".format(len(input_files)))
time.sleep(10)

```

Figure 3.18: Pretraining Script

A brief section of our Python script is also shown in the above figure, with the goal of running our model on a single TPU. Here, we could further double-check that all of the parameters that BERT requires have been correctly defined. We set our model to run for 1 million training steps, similar to Devlin [15], with batches of 256 sequences of length 512 and a learning rate of 1e-4. We even followed BERT's advice and set the maximum predictions to 75 and the probability of masked language modeling to 0.15. We also defined paths to our bucket's folders in GCS and, of course, to our model's vocabulary in addition to these hyperparams.

#### 3.3.2 Pretraining Procedure

Firstly, we have to highlight that in order to run every single time our script, we had to connect with Google Cloud Console via ssh to our VM and ensure that our VM's and TPU's state was nothing but *healthy* and *running*. Moreover, consider that every 20.000 steps, a checkpoint of our model is stored to our bucket, ensuring that we are constantly updating the state of our model.

```

2021-11-07 02:33:19,865 : Outfeed finished for iteration (1, 19638)
2021-11-07 02:33:24,080 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:33:54,157 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:34:20,111 : Outfeed finished for iteration (1, 19754)
2021-11-07 02:34:24,202 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:34:54,283 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:35:20,360 : Outfeed finished for iteration (1, 19870)
2021-11-07 02:35:24,325 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:35:54,383 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:36:20,605 : Outfeed finished for iteration (1, 19986)
2021-11-07 02:36:24,450 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:36:28,258 : Saving checkpoints for 980000 into gs://greek_legal_bert_v2/model/model.ckpt.
2021-11-07 02:36:39,502 : loss = 0.72376215, step = 980000 (10404.044 sec)
2021-11-07 02:36:39,503 : global_step/sec: 1.92233
2021-11-07 02:36:39,503 : examples/sec: 492.116
2021-11-07 02:36:39,504 : Enqueue next (20000) batch(es) of data to infeed.
2021-11-07 02:36:39,504 : Dequeue next (20000) batch(es) of data from outfeed.
2021-11-07 02:36:54,523 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:37:20,818 : Outfeed finished for iteration (2, 77)
2021-11-07 02:37:24,576 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:37:54,625 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:38:21,061 : Outfeed finished for iteration (2, 193)
2021-11-07 02:38:24,699 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:38:54,773 : TPUPollingThread found TPU b'greek-legal-bert-v2-tpu' in state READY, and health HEALTHY.
2021-11-07 02:39:21,305 : Outfeed finished for iteration (2, 309)

```

Figure 3.19: Pretraining Snapshot

Easily, from the above figure, we drive home the point that our TPU v3-8 was capable of completing **116 steps** in just a single minute. As a result, we can claim that our entire training procedure will last 6 days of nonstop running. However, we must underline a key observation, which is that Google disconnects our VM every 12 hours, and these disconnections unambiguously slowed our training method, inferentially delaying model training to the point where our precise model training took **10 days**.

### 3.3.3 Save model

So we've reached the point where BERT pretraining with our legal data is complete, and our GreekLegalBERT v2 is ready to take on downstream tasks. But before moving on, we need to be accurate and careful, in order to save our model properly.








 bert_config.json	me	7 Nov 2021	459 bytes
 bert-vocab.txt	me	7 Nov 2021	497 KB
 model.ckpt-1000000.data-00000-of-00001	me	7 Nov 2021	1.27 GB
 model.ckpt-1000000.index	me	7 Nov 2021	9 KB
 model.ckpt-1000000.meta	me	7 Nov 2021	4.4 MB
 pytorch_model.bin	me	12 Nov 2021	433.2 MB
 tf_to_pytorch.ipynb	me	29 Nov 2021	8 KB

Figure 3.20: Saved files

Initially, we started by retrieving our BERT **configuration file** and **vocabulary** from our bucket, both of which are critical for our ongoing tasks. Apart from those files, we had to obtain our **latest model's checkpoint**, which has obviously as extension, as indicated in the above figure, the final step of pretraining, which is equal to 1 million.

*PyTorch* is an open source machine learning library that specializes in tensor computations, automatic differentiation, and GPU acceleration. PyTorch is one of the most popular deep learning libraries as a result of these factors, competing alongside *Keras* and *TensorFlow* for the title of "most utilized" deep learning package.



Finally, we must highlight that, since we'll be using **Pytorch** instead of Tensorflow in our fine tuning later, we developed a very useful script that takes the previously mentioned files as input and converts a model from Tensorflow to Pytorch.

### 3.3.4 Upload model

*Hugging Face* is a noteworthy open community that has rapidly evolved into an attractive hub for pre-trained deep learning models, particularly for NLP. They are constructing the world's largest collection of models, datasets, and metrics on the Hugging Face Hub in order to democratize and progress AI for everyone.



Figure 3.21: Hugging Face

As a result, anyone can publish their model on Hugging Face Hub and make it available to everyone. This upload was meant to make our job more convenient for us. Each time we perform one of our upcoming fine tuning tasks, we will need to load a distinct BERT model. As a result, we can obtain our model in a single line from the moment we upload it to **Hugging Face Hub**.

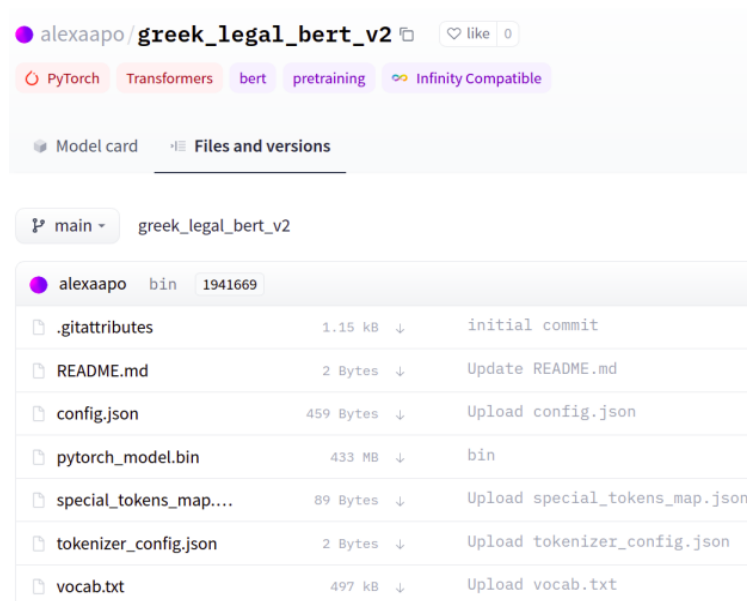


Figure 3.22: Model Card

Except of the configuration file, the vocabulary and our pytorch model we needed to upload a simple file, called special tokens map with extension .json, which contained the 4 characteristic tokens that only BERT understands (SEP, UNK, CLS, PAD).

It should be mentioned that Greek-Bert was already published to Hugging Face Hub, and that we followed the same "uploading" method with the Greek-Legal-BERT-v1 model.

```
from transformers import AutoModel

Greek_Legal_BERT_v1_model = AutoModel.from_pretrained(
    'alexaapo/greek_legal_bert_v1')

Greek_Legal_BERT_v2_model = AutoModel.from_pretrained(
    'alexaapo/greek_legal_bert_v2')

Greek_BERT_model = AutoModel.from_pretrained(
    'nlpaueb/bert-base-greek-uncased-v1')
```

And subsequently, we can from now on load whichever model painless and let our downstream tasks begin!



## 4. TASKS OF GREEKLEGALBERT V2 MODEL

### 4.1 Named Entity Recognition

NER, short for, Named Entity Recognition is a standard Natural Language Processing problem which deals with information extraction. The primary objective is to locate and classify named entities in text into predefined categories such as the names of persons, organizations, locations, events, expressions of times, quantities, monetary values, percentages, etc. Here is an example, where a model identified 4 different entities in a single sentence!

«Ousted **WeWork** founder **Adam Neumann** lists his **Manhattan** penthouse for **\$37.5 million**.»

**Organization**

**Person**

**Location**

**Monetary Value**

#### 4.1.1 Dataset

Our dataset [9] was provided by Nomothesi@ Platform [10]. So this current dataset contains totally 254 text and annotated files. However, this isn't the suitable input format for our NER task, therefore we'll must go through a transformation process to get the necessary format, which is CoNLL (Computational Natural Language Learning).

```
T33 LEG-REFS 0 32 ΠΡΟΕΔΡΙΚΟ ΔΙΑΤΑΓΜΑ ΥΠ' ΑΡΙΘΜ. 22
T52 FACILITY 53 78 Ιερού Ναού Αγίου Λεοντίου
T63 LOCATION-UNK 84 103 περιοχή Παγοποιείου
T39 GPE 109 139 Δήμου Αγίου, του Νομού Αχαΐας
T9 ORG 145 188 Ιεράς Μητροπόλεως Καλαβρύτων και Αιγιαλίας
T61 GPE 205 226 ΕΛΛΗΝΙΚΗΣ ΔΗΜΟΚΡΑΤΙΑΣ
T91 LEG-REFS 284 295 ν. 590/1977
T59 ORG 339 360 Εκκλησίας της Ελλάδος
T60 ORG 468 489 Εκκλησίας της Ελλάδος
T79 LEG-REFS 654 666 π.δ. 63/2005
T12 PUBLIC-DOCS 701 742 αριθμ. ΣΤ5/5557/19.1.2009 κοινής απόφασης
T22 ORG 777 810 Εθνικής Παιδείας και Θρησκευμάτων
T6 ORG 851 895 Υπουργείου Εθνικής Παιδείας και Θρησκευμάτων
T10 ORG 961 1004 Ιεράς Μητροπόλεως Καλαβρύτων και Αιγιαλίας
T54 PUBLIC-DOCS 1006 1029 πρακτικό 1462/31.7.2008
T14 ORG 1055 1092 Δημοτικού Συμβουλίου του Δήμου Αγίου
```

Figure 4.1: Example from Annotated File

More specifically, we first should extract only the information that we care from each annotated file, namely the entity type, the second word, and the last phrase denoting that these exact words are associated to our previously specified entity, and save it in CSV files.

*Text File + Annotated File => CoNLL File*

According to IOB-format, we managed to generate our new files with CoNLL format. We read word by word each text file and its corresponding CSV file, in order to match each word with a single entity, and record a word and its entity in each line every time with the following strategy:

- O, the word doesn't match with any entity.
- I-SomeEntity, the word match with SomeEntity and it is located inside of the sentence (not at the beginning).
- B-SomeEntity, the word match with SomeEntity and it is located at the beginning of the sentence.

```

Σύμφωνα O
με O
το O
Άρθρο O
13 O
του O
Νόμου O
αυτού O
αρμόδια O
αρχή O
στη O
Δημοκρατία B-GPE
της I-GPE
Μολδαβίας I-GPE
για O
την O
ανανεώσιμη O
ενέργεια O
είναι O
η O
Υπηρεσία B-ORG
Ενεργειακής I-ORG
Αποδοτικότητας I-ORG
. O

Η O
Μολδαβική O
πλευρά O
δήλωσε O
ότι O
θα O
μπορούσε O
να O
υπογραφεί O
ένα O
Μνημόνιο O
Κατανόησης O
μεταξύ O
του O
Κέντρου B-ORG
Ανανεώσιμων I-ORG
Πηγών I-ORG
Ενεργειας I-ORG
ΚΑΠΕ I-ORG
της O
Ελληνικής B-GPE
Δημοκρατίας I-GPE
και O
της O
Υπηρεσίας B-ORG
Ενεργειακής I-ORG

```

So on the leftside Figure we have adduced a small portion of a final CoNLL file. As we can observe each line, indeed, contains one unique word of a sentence and its entity type.

BERT does not expect a single unique word and its entity as input; instead, it expects a sentence composed of discrete words and a corresponding sequence of labels, which in our scenario are our entities. As an outcome, our NER dataset parser will differentiate sentences with an extra newline character in the forward.

In regards of our entities, we must declare that there are a total of eight. We've underlined them carefully below:

- Legal References
- Organization
- Public Documentations
- Person
- GeoPolitical Entity
- Location Unknown
- Location National
- Facility

Consider that because of the IOB format, each of our above entities will have different kinds of entities: I-SomeEntity and B-SomeEntity, besides the special O entity. As a result, our final model for NER task will be composed of our BERT model with a dropout layer on top of it, as well as an additional linear layer that outputs vectors with dimension size equal to our total entities, which is 17.

**Figure 4.2: Example from CoNLL file**  
A. Apostolopoulou - S. Briakos

Table 4.1: NER Dataset Statistics

Dataset	Files	Txt Format (words)	Ann Format (words)	CoNLL Format (words)
Train	162 (64%)	409.831	23.723	450.149
Validation	45 (18%)	108.081	5.478	118.032
Test	47 (18%)	103.546	5.089	112.546
Total	254 (100%)	621.458	34.290	680.727

To thoroughly inspect our dataset, we display the above table. Please keep in mind that our entire dataset is roughly 10 MB in terms of capacity and that it has already been splitted into sets. Someone would argue that the quantity of words in CoNLL and text files must be equal, however consider that in CoNLL files when a sentence ends, we insert a new line character to discriminate sentences, justifying the difference in this way.

Our NER dataset isn't quite ready yet, so we'll have to step back a bit before we start fine-tuning our models. This minor imperfection is due to a few sentences that have an unusually large number of words while also containing information such as numerical data or words from the English language. As a result, it's clear why our models generate sequences with more words than BERT's maximum input sequence length of 512 during tokenization. To mitigate this issue, we manually split each sentence into chunks, which we then feed as input into our models. Because of BERT's limitation, we decided to keep the chunked information instead of discarding them.

At the last stage of NER dataset's preprocessing, we've reached a common roadblock when utilizing pre-trained models for token-level classification: many of the tokens in the corpus aren't included in our model's vocabulary. The Twitter handle @huggingface, for example, would be broken into the tokens ['@', 'hugging', '##face'] utilizing BERT's tokenizer. This is a challenge for us because each token has precisely one tag. We'll end up with a mismatch between our tokens and our labels if the tokenizer splits a token into many sub-tokens.

One way to overcome this issue is to **only train on the tag labels for the first subtoken of a split token**. We can do this in Transformers by setting the labels we wish to ignore to -1. In the example above, if the label for @HuggingFace is 3 (indexing B-corporation), we would set the labels of ['@', 'hugging', '##face'] to [3, -1, -1].

### 4.1.2 Fine Tuning

We were able to retrieve and use the bulk of the Greek-BERT code [11], which is written totally in PyTorch, although we had to make a few significant adjustments to meet our requirements for our ongoing fine tuning [23].

During fine tuning for each model, we will experiment with the hyperparameters specified above:

**Table 4.2: NER Fine Tuning Hyperparameters**

Hyperparameter	Value
<b>Learning Rate (Adam)</b>	{2e-5, 3e-5, 5e-5}
<b>Dropout Probability</b>	{0, 0.1, 0.2}
<b>Gradient Accumulation Steps</b>	{2, 4}
<b>Batch Size</b>	8

Devlin [15] and its team highly urge experimenting with the learning rate values, combined with Adam optimizer, that we choosed for fine tuning procedure. However, they recommended that dropout probability be kept at 0.1 at all circumstances, but we opted to experiment a little more in that area. Furthermore, they recommend a batch size of 16 or 32, but due to a lack of computing power, notably memory restrictions, we always choose batch size of 8 and experiment with gradient accumulation steps of 2 or 4.

But how does this strategy address the issue of inconsistency? We can imitate a greater batch size by accumulating gradients. So, in our example, if we wish to use 32 sequences in a single batch, but our hardware only permits us to use 8, our hardware will collapse. In such circumstance, we can use 8-sequence batches and update weights every 4 batches. If we accumulate the gradients from each batch in between, the results will be nearly identical to those obtained by setting the batch size to 32.

Finally, we have in our hands the optimal hyperparameters for each model and thus we can evaluate all models on the test set.

**Table 4.3: NER Fine Tuning Best Hyperparameters**

Model	Macro-F1 Score	Learning Rate	Dropout Probability	Gradient Accumulation Steps	Batch Size
<b>Greek-Legal-BERT-v1</b>	0.71	2e-05	0.1	4	8
<b>GreekLegalBERT v2</b>	0.73	2e-05	0.1	4	8
<b>Greek-BERT</b>	0.74	3e-05	0.2	4	8

Early stopping was being used in each experiment based on the macro-F1 score, which implies that if validation loss decreased from epoch to epoch, training was stopped immediately. After 18 experiments for each model, we were able to identify the optimal hyperparameters for each model in a greedy manner based on the macro-F1 score, as shown in the table above.

## 4.2 Natural Language Inference

Natural Language Inference (NLI) is a task in NLP where we are given two sentences namely Premise and Hypothesis. In this task we are supposed to make a prediction whether the given Hypothesis is **True**, **False** or **Neutral** associated with the Premise. For True we call it Entailment, for False Contradiction and Neutral we use it when it is not related or not determined well.

Table 4.4: NLI Sentence Example

Label	Premise	Hypothesis
<b>Entailment</b>	Και είπε, Μαμά, έφτασα στο σπίτι.	Είπε στην μαμά του ότι είχε πάει σπίτι.
<b>Contradiction</b>	Και είπε, Μαμά, έφτασα στο σπίτι.	Δεν είπε ούτε λέξη.
<b>Neutral</b>	Και είπε, Μαμά, έφτασα στο σπίτι.	Τηλεφώνησε στην μαμά του μόλις το σχολικό λεωφορείο τον άφησε.

### 4.2.1 Dataset

For the train dataset of fine-tuning we used the **Multi-Genre Natural Language Inference (MultiNLI)** [27] corpus, which is a crowd-sourced collection of 393 thousand sentence pairs annotated with textual entailment information. This dataset is also used for fine-tuning the **Greek-BERT** model. The pairs are translated into 14 languages and of course we used only the Greek pairs of them. However, the bad news is that the pairs are automatically (machine)-translated from English to other languages. For this reason, we can easily doubt the quality of this dataset and we can support the fact that this may harm the performance of the model.

The good news is that in Greek-BERT paper [21], they mentioned that they tested the Greek-BERT model with two different training sets:

- Get only the high-quality pairs, approximately 10% of full training set
- Get the full training dataset, including many noising pairs

However, they eventually noticed that the performance using the entire training set was better than the performance training only on the high-quality training subset. So the initial assumption of noisy data could harm the performance, turned out false. In fact, they attributed this phenomenon to the fact that noise may be acting as a regularizer, improving the generalization ability of the model. So we decide that is a good choice to use the whole training set in fine-tuning.

For development and test dataset we used the **Cross-Lingual Natural Language Inference Corpus (XNLI)** [14], which contains:

- 5.000 test pairs
- 2.500 development pairs

for the MultiNLI corpus. They are translated manually by professional translators in order to ensure that the context of the original text is the same as the translated one, with no unnecessary additions or significant cuts that misrepresent the original meaning.

Another issue that concerned us is that we trained our model in stricted Legal data, which is a very specific domain and we are going to test it in data of general domain. XNLI contains sentences of various content, such as face-to-face conversations, letters, telephone speeches, reports, fictions etc. Under those circumstances, we consider logic the fact that our model will not have a good performance in this task with this dataset. It could be more interesting if we could test it in a specific-domain dataset (based mainly on legal data), but there isn't such a dataset available yet.

Consequently, we make an initial prediction of a not-so-satisfactory performance of our model. Nevertheless, it is still a challenge to see if our assumption is right or not. Above they are specified the total dataset statistics:

Table 4.5: NLI Dataset Statistics

Dataset	Pairs	Ratio	Capacity
Training (MultiNLI)	392.702	98.13%	141,8 MB
Development (XNLI)	2.490	0.62%	848,5 KB
Test (XNLI)	5.010	1,25%	1,7 MB
Total	400.202	100%	144,4 MB

## 4.2.2 Fine Tuning

For implementing NLI, we used a vocabulary and token-index mapping in order to let the model understand our inputs. The whole vocabulary size is 35.000. Let us adduce an example to comprehend how GreekLegalBert v2 tokenizer works:

```
tokens = tokenizer.tokenize('Και είπε, Μαμά, έφτασα στο σπίτι.')
print(tokens)

['και', 'ειπε', ',', 'μαμα', ',', 'εφτα', '##σα', 'στο', 'σπιτι', '.']
```

Figure 4.3: GreekLegalBERT v2 Tokenizer Example

In such a form, tokens can be easily converted to index:

```
indexes = tokenizer.convert_tokens_to_ids(tokens)
print(indexes)

[414, 2887, 16, 22311, 16, 11798, 598, 476, 8283, 18]
```

Figure 4.4: GreekLegalBERT v2 Indexes Example

We also need to give input to the Legal BERT in the same format in which BERT has been pre-trained. BERT uses two special tokens denoted as [CLS] and [SEP], as previously stated. For example:

- **Premise:** Και είπε, Μαμά, έφτασα στο σπίτι.
- **Hypothesis:** Δεν είπε ούτε λέξη.
- **Input Format:** [CLS] Και είπε, Μαμά, έφτασα στο σπίτι. [SEP] Δεν είπε ούτε λέξη. [SEP]



After 18 experiments for each model, we were able to identify the optimal hyperparameters for each model in a greedy manner based on the macro-F1 score, as shown in the table below:

**Table 4.7: NLI Fine Tuning Best Hyperparameters**

<b>Model</b>	<b>Macro-F1 Score</b>	<b>Learning Rate</b>	<b>Dropout Probability</b>	<b>Gradient Accumulation Steps</b>	<b>Batch Size</b>
<b>Greek-Legal-BERT-v1</b>	0.65	3e-05	0.1	2	8
<b>GreekLegalBERT v2</b>	0.70	2e-05	0	4	8
<b>Greek-BERT</b>	0.73	2e-05	0	4	8

### 4.3 Multiclass Classification on Raptarchis

To prevent misinterpretation, we must clarify that only the Raptarchis dataset's train set was provided to our model's pretraining process in this task, as we previously described in Section 3.1.

#### 4.3.1 Dataset

We believe we can concatenate the development and test sets and break them again into train, development, and test sets since we have previously utilized the train set during pre-training of GreekLegalBERT v2 model. We'll have all the datasets we ought to fine-tune our model and produce the desired predictions this way.

**Table 4.8: Capacities of remaining Raptarchis**

<b>Dataset</b>	<b>Items</b>	<b>Capacity</b>
Development	9.512	73 MB
Test	9.516	76 MB
Total	19.028	149 MB

We re-split the old development and test sets as follows:

**Table 4.9: Capacities of new fine tuning Raptarchis dataset**

<b>Dataset</b>	<b>Items</b>	<b>Ratio</b>	<b>Capacity</b>
Train	15.222	80%	119 MB
Development	1.903	10%	15 MB
Test	1.903	10%	15 MB
Total	19.028	100%	149 MB

As far as we can discern, we can only experiment with the all labels category with this new split up. Because we used the Raptarchis Dataset as the initial training set for GreekLegalBERT v2 pretraining, it's understandable that we won't have all of the labels for each category. Let's take a closer look at the table below:



**Table 4.10: Difference of Labels between Raptarchis datasets**

<b>Set</b>	<b>Volume Labels</b>	<b>Chapter Labels</b>	<b>Subject Labels</b>
Original Raptarchis	47	389	2285
New Resplit Raptarchis	47	383	1970

To summarize, we are going to experiment with 100% of Volume labels, with 98% of Chapter labels and with 86% of Subjects labels compared to initial Raptarchis dataset. For each category, we can now proceed to three discrete multiclass classification tasks.

### 4.3.2 Fine Tuning

Based on Vamvourellis [26] thesis, we are going to experiment with his suggested pre-defined hyperparameters, which are specified above:

**Table 4.11: Multiclass Classification on Raptarchis Fine Tuning Hyperparameters**

<b>Hyperparameter</b>	<b>Value</b>
<b>Learning Rate (Adam)</b>	2e-5
<b>Dropout Probability</b>	0.1
<b>Max Sequence Length</b>	512
<b>Batch Size</b>	8
<b>Epochs for Volume:</b>	5
<b>Epochs for Chapter:</b>	9
<b>Epochs for Subject:</b>	16

We were capable of obtaining scores from each category classification for each model after a series of experiments with multiple seed values. Each epoch in every classification task lasts 10 minutes, therefore classifying all categories for our three discrete models required a total of 14 hours.

## 4.4 Performance on Masked Language Modeling (MLM)

During pretraining, the model was trained with two different tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), both of which were thoroughly detailed in Chapter 2. So, just before we demonstrate our models' performance on our downstream tasks, now seems to be the time to cross-check our model's performance in MLM by examining the examples below.

---

Example 1

---

**Input text:** Ο [MASK] προσανατολισμός της νέας φαρμακευτικής πολιτικής διατρέχει το σύνολο των επί μέρους διατάξεων του νόμου.

Model's Answer 1: γενικός

Model's Answer 2: βασικός

Model's Answer 3: άλλος

**Correct Answer:** κοινωνικός

---

Example 2

---

**Input text:** Η [MASK] ενός ταμείου που διευκολύνει την κίνηση του πετρελαίου σ' όλη τη χώρα.

Model's Answer 1: λειτουργία

Model's Answer 2: διοίκηση

Model's Answer 3: δημιουργία

**Correct Answer:** δημιουργία

---

Example 3

---

**Input text:** Οι κανόνες [MASK] των δεδομένων προσωπικού χαρακτήρα διέπουν σημαντικές πτυχές του τρόπου αλληλεπίδρασης των επιγραμμικών υπηρεσιών με τους χρήστες.

Model's Answer 1: προστασίας

Model's Answer 2: επεξεργασίας

Model's Answer 3: διαχείρισης

**Correct Answer:** προστασίας

---

Example 4

---

**Input text:** Πρέπει, λοιπόν, να προωθήσουμε και να [MASK] με τεράστιες προσπάθειες αυτή την εκπαιδευτική πολιτική.

Model's Answer 1: προωθησουμε

Model's Answer 2: εφαρμόσουμε

Model's Answer 3: στηριξουμε

**Correct Answer:** εδραιώσουμε

---

Example 5

---

**Input text:** Να διασφαλίσουμε ότι έχουμε περισσότερες ευρωπαϊκές ευθύνες εδώ και να εφαρμόσουμε περισσότερες κοινές [MASK] .

Model's Answer 1: δράσεις

Model's Answer 2: θεσεις

Model's Answer 3: αποφασεις

**Correct Answer:** δράσεις

---

In the previous examples, we demonstrate 5 consecutive instances in which we feed our model a specific input text and provide the three answers that the model believes are close to the correct answer. We attempted to have a wide range of inputs in terms of MASK's position inside the input sentence (at the start, middle, or end), as well as force our model to predict various parts of speech words, such as nouns, verbs, and adjectives.

In particular, in Examples 3 and 5, our GreekLegalBERT v2 confidently predicted the correct as its first answer, whereas in Example 2, he predicted the correct as its third answer. However, our model was unable to predict the desired word in the remaining cases. This observation does not seem to be so discouraging, mainly due to the fact that GreekLegalBERT v2 predicted several words that could clearly be expressed in a Greek well-structured phrase in both of the "wrong" Examples. As a result, the model performed admirably in terms of semantics.

## 4.5 Results

Before we proceed any further, it's essential to highlight a few key strategies that were used in each of our 3 distinct tasks.

- So, in order to represent the forthcoming outcomes for each task, we must give special credits to our supervisors, who let us to utilize a single **GPU** of type GeForce RTX 2080 Ti, which belongs to AI team of Department of Informatics and Telecommunications of University of Athens. This pre-declared infrastructure enabled us to actually accelerate our fine tuning experiments without any network constraints, resulting in a more than safe and consistent environment in which we run all BERT models.
- Moreover, we applied another similar technique in all of our tasks, which was **early stopping**. It's the process of retaining information about train and validation data as a model is learning. Specifically we keep a record of the loss function on the validation data, and stopping rather than going through all the epochs when we detect no improvement on the validation set.
- Ultimately, we'll go over the **metrics** that were used to evaluate all our models. Initially through a very simple table we will give a rough sense about True Positives, True Negatives, False Positives, False Negatives.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 4.7: True Positives-Negatives, False Positives-Negatives

Once we understand these four parameters then we can calculate *Precision*, *Recall* and *F1 score*.

$$Precision = \frac{True\ positive}{True\ positive + False\ Positive}$$

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}$$

$$F1 = 2 * \frac{Precision + Recall}{Precision * Recall}$$

#### 4.5.1 Named Entity Recognition

These scores were computed per *token*, using an evaluation per entity technique that enables for a more precise assessment of each model's performance and finally represent micro, macro and average scores. The final results, which are the average of 5 identical model executions, are described in the table below, with the *standard deviation* as a percentage marked by the numbers in parentheses:

**Table 4.12: NER Results**

Entity Type	Greek-BERT			Greek-Legal-BERT-v1			GreekLegalBERT v2		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
FACILITY	0.24 (6%)	0.22 (7%)	0.23 (6%)	0.20 (3%)	0.20 (4%)	0.20 (3%)	0.27 (1%)	0.27 (1%)	<b>0.27 (0%)</b>
GPE	0.68 (2%)	0.74 (1%)	<b>0.71 (1%)</b>	0.69 (2%)	0.72 (2%)	<b>0.71 (2%)</b>	0.69 (1%)	0.73 (0%)	<b>0.71 (1%)</b>
LEG-REFS	0.80 (2%)	0.83 (0%)	<b>0.81 (1%)</b>	0.79 (0%)	0.82 (1%)	<b>0.81 (1%)</b>	0.79 (1%)	0.82 (0%)	<b>0.81 (1%)</b>
LOCATION-NAT	0.79 (11%)	0.78 (4%)	<b>0.78 (7%)</b>	0.78 (6%)	0.60 (6%)	0.67 (4%)	0.83 (4%)	0.53 (2%)	0.65 (3%)
LOCATION-UNK	0.52 (2%)	0.61 (1%)	0.56 (2%)	0.54 (4%)	0.62 (3%)	0.58 (4%)	0.57 (2%)	0.64 (3%)	<b>0.61 (2%)</b>
ORG	0.63 (2%)	0.76 (2%)	0.69 (2%)	0.66 (1%)	0.76 (2%)	<b>0.71 (1%)</b>	0.66 (1%)	0.78 (0%)	<b>0.71 (1%)</b>
PERSON	0.80 (2%)	0.81 (1%)	0.80 (1%)	0.80 (3%)	0.82 (3%)	<b>0.81 (3%)</b>	0.79 (1%)	0.81 (0%)	0.80 (0%)
PUBLIC-DOCS	0.68 (4%)	0.73 (11%)	0.70 (7%)	0.68 (2%)	0.73 (3%)	0.70 (1%)	0.69 (1%)	0.75 (1%)	<b>0.71 (1%)</b>
Micro AVG	0.69	0.76	0.72	0.70	0.75	0.73	0.70	0.76	0.73
Macro AVG	0.64	0.68	0.66	0.64	0.66	0.65	0.66	0.67	0.66
Weighted AVG	0.69	0.76	<b>0.72</b>	0.70	0.75	<b>0.72</b>	0.70	0.76	<b>0.73</b>

So we've finally reached a point where the outcomes of our models are more than ready. We have already highlighted the strongest F1 score in each entity type in order to distinguish the winner model. We can clearly see all three models achieve equal scores in terms of GeoPolitical and Legal References entities. In terms of the Organization Entity, there is also a tie between the Greek-Legal-BERT models. Moreover, Greek-BERT outperformed in the Location-Nationality entity. Finally, our model is the only victor in all remaining entities except Person's, where Greek-Legal-BERT-v1 slightly outperforms. Overall, our model wins with a 0.1 difference compared between the two remaining models, receiving a weighted average F1 Score of **0.73**. With this final result, we may be able to justify why our new Greek-Legal-BERT version outperformed, due to the increase in input Legal dataset size from 4.5GB to 8.6GB.

## 4.5.2 Natural Language Inference

These scores were computed per *class*, using an evaluation per class technique that enables for a more clear measurement of each model's performance and finally represent F1 accuracy macro and average scores. The final results, which are the average of 5 identical model executions, are described in the table below, with the *standard deviation* as a percentage marked by the numbers in parentheses:

**Table 4.13: NLI Results**

Class	Greek-Legal-BERT-v1			GreekLegalBERT v2			Greek-BERT		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
ENTAILMENT	0.59 (1%)	0.69 (3%)	0.63 (1%)	0.62 (1%)	0.76 (2%)	0.68 (1%)	0.67 (3%)	0.74 (4%)	<b>0.70 (1%)</b>
CONTRADICTION	0.67 (4%)	0.68 (4%)	0.67 (1%)	0.71 (1%)	0.72 (3%)	0.71 (1%)	0.71 (2%)	0.78 (2%)	<b>0.74 (1%)</b>
NEUTRAL	0.73 (2%)	0.58 (2%)	0.64 (1%)	0.78 (1%)	0.59 (2%)	0.67 (1%)	0.82 (2%)	0.64 (4%)	<b>0.72 (2%)</b>
Accuracy			0.65			0.69			0.72
Macro AVG	0.66	0.65	0.65	0.70	0.69	0.69	0.74	0.72	0.72
Weighted AVG	0.66	0.65	<b>0.65</b>	0.70	0.69	<b>0.69</b>	0.74	0.72	<b>0.72</b>

As we can see, our initial predictions for the performance of our model, as well as the other models, have been verified. From the beginning, the comparison was a little unjust. Greek-BERT model outperforms for all classes and it has the best weighted average F1 of 0.72%. This is justified by the fact that we tested general content data in a general purpose model. Nevertheless, our initial goal, to overcome the scores of the first Greek Legal model achieved, as our model surpassed it individually in all three classes, but also as a weighted average F1, after catching 0.69% score against 0.65%. As previously said, if a Greek legal dataset for the NLI task existed, it would be more interesting to see if our model performs better with data similar to those used to train it. However, we believe that if such a dataset ever exists, it will outperform even the Greek-Model score, since even now the disparity in scores isn't really large.

## 4.5.3 Multiclass Classification on Raptarchis

Finally we adduce carefully our last task's results, which are based on Precision, Recall and F1 score.

**Table 4.14: Multiclass Classification on Raptarchis Results**

Model	Volume Category			Chapter Category			Subject Category		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Greek-BERT	84.50	84.50	84.50	76.83	76.83	76.83	72.83	72.83	72.83
Greek-Legal-BERT-v1	85.50	85.50	85.50	79.51	79.51	79.51	75.51	75.51	<b>75.51</b>
GreekLegalBERT v2	87.76	87.76	<b>87.76</b>	81.40	81.40	<b>81.40</b>	74.93	74.93	74.93

The results of this task are considerably simpler than those of the previous tasks, which is justified because we documented the same value in Precision, Recall, and F1 score in each experiment of each model for each category. Furthermore, scores are fairly evident because we have an unambiguous victor in each category, notably our model for the Volume and Chapter categories and the first version of Greek-Legal-BERT for the Subject category.

## 5. CONCLUSIONS AND FUTURE WORK

At the beginning of this thesis, we studied the architecture of the BERT model and see how exactly it works. We also explored five distinct Greek legal datasets that we used to train our model.

Then we started working on our own BERT model, which was based on Greek Legal Data. We walked you through all of the preparatory procedures, from file pre-processing to creating Vocabulary and TensorFlow Records, until we were ready to train our model. We demonstrated how the Google Cloud Platform helped us train for 10 days straight. To solidify our endeavor, we tested, evaluated and compared our model test against Greek-BERT and Greek-Legal-BERT-v1 on three separate tasks: NER, NLI, and Multiclass Classification on Raptarchis.

Our first goal was to improve on the outcomes of the earlier edition of Greek-Legal-BERT and compare it to GreekBERT, a general-purpose heavy-trained model. In any case, we accomplished our initial target, since we defeated the Greek-Legal-BERT-v1 in all three tasks. In terms of Greek-BERT, our model outperforms it in two of three tasks in general, with the exception of NLI, which is logical given the fact that we tested with a dataset of general data in a stricted-legal domain model.

For the future, except for Legal Data, we intend to pre-train BERT-based models for earlier forms of Greek, particularly classical Greek, for which large datasets are available [6] [7]. This could potentially lead to improved NLP tools for classical studies and not only.

## ABBREVIATIONS - ACRONYMS

AI	Artificial Intelligence
BERT	Bidirectional Encoder Representations from Transformers
LegalAI	Legal Artificial Intelligence
VM	Virtual Machine
NLP	Natural Language Processing
EU	European Union
GCP	Google Cloud Program
GCE	Google Compute Engine
GCS	Google Cloud Storage
TPU	Tensor Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
TFRC	Tensorflow Research Cloud
MLM	Masked-Language Modeling
NER	Named Entity Recognition
NLI	Natural Language Inference
NSP	Next Sentence Prediction
TFRecord	Tensorflow Record
IOB	Inside – Outside – Beginning
GPE	GeoPolitical Entity
LEG-REFS	Legal References
LOCATION-NAT	Location-National
LOCATION-UNK	Location-Unknown
ORG	Organization
PUBLIC-DOCS	Public-Documents
AVG	Average
SSH	Secure Shell
UTF-8	Unicode Transformation Format 8 bit
CoNLL	Computational Natural Language Learning

CSV	Comma Separated Values
KB	Kilo Bytes
MB	Mega Bytes
GB	Giga Bytes
URL	Uniform Resource Locators
MultiNLI	Multi-Genre Natural Language Inference
XNLI	Cross-Lingual Natural Language Inference Corpus



## REFERENCES

- [1] BERT Explained: A Complete Guide with Theory and Tutorial. <https://towardsml.com/2019/09/17/bert-explained-a-complete-guide-with-theory-and-tutorial/>.
- [2] EurLEX. <https://eur-lex.europa.eu/homepage.html>.
- [3] EuroVoc. <https://op.europa.eu/en/web/eu-vocabularies>.
- [4] Greek Wikipedia. <https://dumps.wikimedia.org/elwiki/>.
- [5] Hellenic Parliament Sessions. <https://www.hellenicparliament.gr/Praktika/Synedriaseis-0lomeleias>.
- [6] Perseus Digital Library. <https://www.perseus.tufts.edu/hopper/>.
- [7] Thesaurus Linguae Graecae. <http://stephanus.tlg.uci.edu/>.
- [8] Rahul Agarwal. Explaining BERT Simply Using Sketches, 2021. <https://mlwhiz.medium.com/explaining-bert-simply-using-sketches-ba30f6f0c8cb>.
- [9] Iosif Angelidis, Ilias Chalkidis, and Manolis Koubarakis. Named Entity Recognition, Linking and Generation for Greek Legislation, 2018. <http://cgi.di.uoa.gr/~koubarak/publications/2018/jurix2018.pdf>.
- [10] Iosif Angelidis, Ilias Chalkidis, Charalampos Nikolaou, Panagiotis Soursos, and Manolis Koubarakis. Nomothesia: A Linked Data Platform for Greek Legislation, 2018. <http://legislation.di.uoa.gr/>.
- [11] Ilias Chalkidis. GREEK-BERT Code, 2020. <https://github.com/nlpaueb/greek-bert>.
- [12] Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. Neural Legal Judgment Prediction in English, 2019. <https://arxiv.org/pdf/1906.02059.pdf>.
- [13] Papaloukas Christos. Greek Legal Code, 2021. [https://huggingface.co/datasets/greek\\_legal\\_code](https://huggingface.co/datasets/greek_legal_code).
- [14] Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel R. Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating Cross-lingual Sentence Representations, 2018. <https://cims.nyu.edu/~sbowman/xnli/>.
- [15] Jacob Devlin, Ming-Wei Changm, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018. <https://arxiv.org/pdf/1810.04805v1.pdf>.
- [16] Thomas Gamauf. Tensorflow Records? What they are and how to use them, 2018. <https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564>.
- [17] Cathal Horan. 10 Things You Need to Know About BERT and the Transformer Architecture That Are Reshaping the AI Landscape, 2021. <https://neptune.ai/blog/bert-and-the-transformer-architecture-reshaping-the-ai-landscape>.
- [18] Chetna Khanna. WordPiece: Subword-based tokenization algorithm, 2018. <https://towardsdatascience.com/wordpiece-subword-based-tokenization-algorithm-1fbd14394ed7>.
- [19] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation, 2005. <https://homepages.inf.ed.ac.uk/pkoehn/publications/europarl-mtsummit05.pdf>.
- [20] Athinaios Konstaninos. Named Entity Recognition using a Novel Linguistic Model for Greek Legal Corpora based on BERT model, 2020. <https://pergamos.lib.uoa.gr/uoa/dl/object/2927727>.
- [21] John Koutsikakis, Ilias Chalkidis, Prodromos Malakasiotis, and Ion Androutsopoulos. GREEK-BERT: The Greeks visiting Sesame Street, 2020. <https://arxiv.org/pdf/2008.12014.pdf>.
- [22] Ben Lutkevich. BERT language model, 2020. <https://searchenterpriseai.techtarget.com/definition/BERT-language-model>.
- [23] Chris McCormick. BERT Fine-Tuning Tutorial with PyTorch, 2019. <https://mccormickml.com/2019/07/22/BERT-fine-tuning/#4-train-our-classification-model>.

- [24] Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. OSCAR, 2019. <https://oscar-corpus.com/>.
- [25] Sarthak Vajpayee. Understanding BERT — (Bidirectional Encoder Representations from Transformers), 2020. <https://towardsdatascience.com/understanding-bert-bidirectional-encoder-representations-from-transformers-45ee6cd51eef>.
- [26] Efstratios Vamvourellis. Comp-BERT-ition: Which BERT model is better for Greek legal text classification?, 2021. <https://pergamos.lib.uoa.gr/uoa/dl/frontend/el/browse/2960898>.
- [27] Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference), 2018. <https://aclanthology.org/N18-1101/>.
- [28] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing, 2020. <https://arxiv.org/pdf/1910.03771.pdf>.
- [29] Haoxi Zhong, Chaojun Xiao, Cunchao Tu, Tianyang Zhang, Zhiyuan Liu, and Maosong Sun. How Does NLP Benefit Legal System: A Summary of Legal Artificial Intelligence, 2020. <https://arxiv.org/pdf/2004.12158.pdf>.
- [30] 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. <https://arxiv.org/pdf/1506.06724.pdf>.