
**ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ**

**ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ**
**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ**

**ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ ΣΤΗΝ ΓΛΩΣΣΙΚΗ ΤΕΧΝΟΛΟΓΙΑ**


**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# DistilBERT-EL-KLD: Απόσταξη Γνώσης και Μοντελοποίηση της Ελληνικής Γλώσσας


**Αθανάσιος Σ. Κουρσάρης**

**Επιβλέπων:** **Μανόλης Κουμπαράκης,** Καθηγητής, ΕΚΠΑ



**ΑΘΗΝΑ**

**ΜΑΡΤΙΟΣ 2024**


**NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS**

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

**INTERDEPARTMENTAL PROGRAM OF POSTGRADUATE STUDIES IN LANGUAGE TECHNOLOGY**


**MASTER'S THESIS**

# DistilBERT-EL-KLD: Knowledge Distillation and Greek Language Modeling


**Athanasios S. Koursaris**

**Supervisor:** **Manolis Koubarakis,** Professor, NKUA



**Athens**

**MARCH 2024**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# DistilBERT-EL-KLD: Απόσταξη Γνώσης και Μοντελοποίηση της Ελληνικής Γλώσσας

**Αθανάσιος Κουρσάρης**
**S.N.:** 7115182100014

**Επιβλέπων:** **Μανόλης Κουμπαράκης,** Καθηγητής, ΕΚΠΑ

**ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:**
**Μανόλης Κουμπαράκης,** Καθηγητής, ΕΚΠΑ
**Ιωάννης Παναγάκης,** Αναπληρωτής Καθηγητής, ΕΚΠΑ
**Σωκράτης Σοφιανόπουλος** Συνεργαζόμενος Ερευνητής, ΙΕΛ

Μάρτιος 2024

**MASTER'S Thesis**

# DistilBERT-EL-KLD: Knowledge Distillation and Greek Language Modeling

**Athanasios S. Koursaris**

**S.N.:** 7115182100014

**Supervisor:** **Manolis Koubarakis,** Professor, NKUA

**EXAMINATION COMMITTEE:**
**Manolis Koubarakis,** Professor, NKUA
**Ioannis Panagakis,** Associate Professor, NKUA
**Sokratis Sofianopoulos,** Associate Researcher, ILSP

March 2024

# ΠΕΡΙΛΗΨΗ

Σκοπός της εν λόγω διπλωματικής εργασίας είναι η ανάπτυξη, εκπαίδευση και αξιολόγηση ενός μοντέλου DistilBERT, εξειδικευμένου αποκλειστικά στην Ελληνική γλώσσα. Μετά από εμβριθή ανασκόπηση του θεωρητικού υποβάθρου (κλασική μηχανική μάθηση, βαθιά μάθηση και νευρωνικά δίκτυα) με σκοπό να καταστεί αντιληπτή η αρχιτεκτονική των νευρωνικών δικτύων της οικογένειας των δικτύων μετασχηματιστών (Transformers) και συγκεκριμένα των μοντέλων BERT και DistilBERT, περιγράφεται αναλυτικά η διαδικασία ανάπτυξης και προεκπαίδευσης του μοντέλου σε μεγάλα σώματα ελληνικών κειμένων (OSCAR, Wikipedia, Europarl) μέσω της διαδικασίας Απόσταξης Γνώσης, καθώς και της περαιτέρω εκπαίδευσης σε διεργασίες φυσικής γλώσσας, όπως η Αναγνώριση Επώνυμων Οντοτήτων (NER), η αυτοματοποιημένη Αναγνώριση Μερών του Λόγου (PoS Tagging), καθώς και η διαδικασία Διεξαγωγής Συμπερασμάτων σε Φυσική Γλώσσα (NLI). Η τεχνική της απόσταξης γνώσης, η οποία αποτελεί μία μορφή συμπίεσης των παραμέτρων ενός νευρωνικού δικτύου, φαίνεται να συμβάλλει καταλυτικά στη δημιουργία μοντέλων βαθιάς μάθησης, τα οποία αν και, συγκρινόμενα με συγγενικά μοντέλα, είναι κατά πολύ ταχύτερα και πιο οικονομικά από υπολογιστική άποψη, δεν παρουσιάζουν μεγάλες απώλειες όσον αφορά την ακρίβεια σε διεργασίες Επεξεργασίας Φυσικής Γλώσσας. Το μοντέλο που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας (DistilBERT-EL-KLD), το οποίο αποτελεί συμπιεσμένη μορφή του GREEK-BERT, έχει τη δυνατότητα να παράγει αποτελέσματα απολύτως συγκρίσιμα με αυτά του προκατόχου του.

**ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ**: Τεχνητή Νοημοσύνη, Επεξεργασία Φυσικής Γλώσσας

**ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ**: συμπίεση, απόσταξη γνώσης, νευρωνικά δίκτυα, βαθιά μάθηση, ταξινόμηση

# ABSTRACT

The purpose of the following thesis is the implementation, pre-training, fine-tuning, and evaluation of a DistilBERT model specialized in the Modern Greek Language. After an in-depth examination of the theoretical background in classical Machine Learning and Deep Learning with Neural Networks with the purpose of understanding the inner workings of Transformer Neural Networks and especially BERT and DistilBERT, what follows is the detailed description of the development process concerning such a model, from its pretraining on large Modern Greek Language Corpora, to its fine-tuning and evaluation on downstream Natural Language Processing Tasks, such as Named Entity Recognition, Part of Speech Tagging, as well as Natural Language Inference. The Knowledge Distillation technique seems to play a paramount role in the creation of models that appear to be faster and relatively computationally inexpensive when compared to other larger similar architectures, yet they seem to exhibit rather insignificant downgrades, if any, when it comes to accuracy. The model developed for the purposes of this thesis (DistilBERT-EL-KLD), being a compressed version of GREEK-BERT, appears to exhibit very similar performance and output results to those of its predecessor.

**SUBJECT AREA**: Artificial Intelligence, Natural Language Processing

**KEYWORDS**: compression, knowledge distillation, neural networks, deep learning, classification

*To Ellie, because living with someone who's writing a master's thesis can be as hard as writing one yourself.*

## ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Professor Manolis Koubarakis whose constant guidance shed light on topics that seemed daunting during the early stages of research. I would also like to thank my family and close friends for showing keen interest in my research and providing their support at every step.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# PROLOGUE

The recent advent of Large Language Models (LLMs) [30] has cemented the ubiquity of AI in a plethora of everyday life scenarios. Even though applications powered by all manner of deep learning models have existed for quite some time (be it smart assistants, machine translation systems or biometric security applications such as face detection for smartphones), harnessing the power of Natural Language Processing has been greatly simplified, since the manner of interaction with (seemingly at least) intelligent general-purpose models can be performed by means of a simple graphical interface, not dissimilar to that of a search engine. Models like GPT-4, Llama 2 and Google Bard have been trained on vast natural language corpora, starting from the task of text generation. Their implementation has proven to entail solutions for more basic problems, like text classification or machine translation, raising the question whether smaller, more-domain specific models have any place in the current landscape in the field of Artificial Intelligence at all.

However, LLMs do not come without significant downsides, pertaining both to their architecture as well as their various implementations per se. Large Language Models dwarf previous NLP models in sheer size of parameters [30]. Even when comparing GPT-4 with previous transformer architectures: when one puts its 1.7 trillion parameters next to BERT's 110 million, the difference is apparent at first glance [4]. This difference in parameter size comes with significant hurdles, such as an exponential growth of requirements both in training time and resources. Other problems rise from the nature of the corpora. Again, the magnitude of said corpora renders LLMs extremely efficient in general-purpose tasks, but sometimes the answer to a given prompt might be fictitious or biased, a phenomenon widely referred to as "hallucination". Domain adaptation also remains a difficult task, due to computational requirements, as well as licensing issues, since not all LLMs have been open-sourced for the time being. Another significant issue with LLMs is the environmental cost of their training and benchmarking as well the massive data storage involved in the process, since it requires the full power of a multitude of GPUs and CPUs, which leads to high levels of electrical draw and heat exhaust [22]. The carbon footprint of LLMs should raise questions concerning the minimization of said requirements.

A more specific problem is the way LLMs handle different languages. Human-like performance can be observed in a multitude of English Language tasks as well as other popular languages, such as Spanish, but when prompted to produce less popular and confined languages such as Greek, the outputs come with significant flaws, stemming directly from the given language's linguistic complexity and various systems pertaining to tenses, cases, figurative speech and much more.

Even so, LLMs have found their place in the industry and have proven their utility on multiple occasions. Nevertheless, there are still ways to achieve human-like performance with much smaller models. Transfer Learning and fine-tuning open-source models on more contained and focused corpora has facilitated overcoming the challenge of domain adaptation in tasks, such as Text Classification, Named Entity Recognition, Machine Translation and more [4]. The DistilBERT model, whose number of parameters is as low as 65 million can be proven a viable solution for these tasks [22]. Trained by distilling the knowledge [10] of the

significantly more-computationally intensive (yet much smaller in size than an LLM) BERT model, DistilBERT is presented as a faster, computationally cheaper and more portable solution, since it can be utilized for inference even in smart devices. What follows is the rigorous description of an attempt to adapt the base-uncased version of the DistilBERT model for the Greek Language by distilling the knowledge of the original GREEK-BERT model, as well as fine-tuning it for a plethora of tasks, such as Part-of-Speech Tagging, Named Entity Recognition and Natural Language Inference and finally, comparing its performance to that of the original GreekBERT [15]. An attempt will also be made to give context regarding the Transformer Neural Network architecture that powers all of the aforementioned models, as well as its place in the larger fields of Deep Learning and AI.

# 1. INTRODUCTION

The following section attempts to provide a high-level overview on the history of the field of Artificial Intelligence and in particular, Machine Learning, mainly focusing on the Deep Learning Revolution as heralded by Artificial Neural Networks and RNN architectures, the latter being the dominant architecture in NLP up to the advent of the Transformers.

## 1.1 Machinery, Intelligence and Cognition

Are machines capable of producing original thought? In the 19th Century, Ada Lovelace's remarks on Charles's Babbage's analytical engine, hailed by some as the world's first mechanical computer, due to its ability to perform simple mathematical operations, present a rather grim and apprehending outlook on the matter: *The Analytical Engine has no pretensions whatever to originate anything* she claims [1]. This outlook would later be challenged by pioneers in the emerging field of computer science, namely Alan Turing, whose Turing Test, appearing as early as 1950 [28], still stands out as one of the first evaluation methods concerning what is now called "Human-like performance". Simplistic, yet quite intuitive, the Turing Test was the first method of determining whether a computer possesses the ability to demonstrate human intelligence and was the first widely known comparison ever made between machine and man.

This was only the beginning, since the publication of Turing's paper *Computing Machinery and Intelligence* was only a handful of years apart from the Dartmouth Workshop, brainchild of Mathematician John McCarthy, considered by many to be the starting point of the field of Artificial Intelligence. In short, the proposal made by the conference is *that every aspect of learning or any feature of intelligence can in principle be so precisely described that a machine can be made to simulate it* [19]. The given goal could not be achieved over the span of a single summer, but the conference drew mass attention and helped plant the seeds of the AI Revolution still experienced today. The first approach regarding the automation of intellectual tasks, as proposed by the Dartmouth Workshop, was hardcoding a vast set of rules that could handle every scenario in said tasks. This approach, today known as *Symbolic AI* [19]*, was* the dominant method until the 1980s and the *expert systems* that came into fruition during the 1980s are closely associated with it. Systems based on symbolic approaches were proven to handle well-defined problems associated with a rigid set of rules, for example simulating games such as chess or Go [6]. Complex problems such as Computer Vision and Natural Language Processing would remain unsolved, until the advent of a new programming paradigm that does away with sheer determinism.

## 1.2 Machine Learning

A possible answer to what Alan Turing calls Lady Lovelace's Objection [28] came in the form of Machine Learning. This new paradigm eschews the traditional approach of writing a program, i.e. designing a set of hand-crafted step-by-step instructions for the computer to follow, in order to produce a desirable output, and instead favors the approach of training a system from data. Instead of rules, the computer is presented with the correct inputs and outputs of a program performing a given task, and utilizes algorithms that enable it to draw patterns and statistical structures from said inputs and outputs [13].

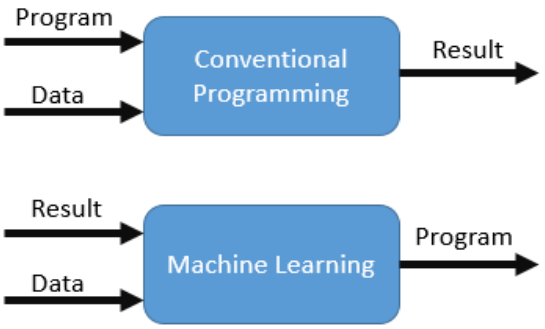


**Figure 1: Comparison between ML and traditional programming**

### 1.2.1 General Overview

A more engineering-oriented definition of the concept is as follows [1]:

*A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.*

Even though algorithms used in Machine Learning have been around since the early days of the field of AI, it was only during the late 1980s and early 1990s that ML approaches started to take off, due to the wider availability of computational resources as well as the easier accessibility to data. To summarize, the key components of each ML system are the following [6]:

- Input data used for training, e.g. movie reviews in a sentiment classification task
- Expected Output data, e.g. a set of labels describing the relative positivity or negativity of said reviews.
- A model used for inference, i.e. whenever the model is given a set of reviews, it produces a set of labels.
- A metric used to evaluate the model's performance, usually a loss function able to measure the degree of mistaken or undesirable inference made by the

model. The main objective of training a ML system is to minimize the loss function.

All ML systems are associated with a life cycle [12], which entails continuous optimization through data collection and processing, model design, training, evaluation and monitoring. High performance is not always guaranteed and the task at hand has to be continuously reframed and reexamined.

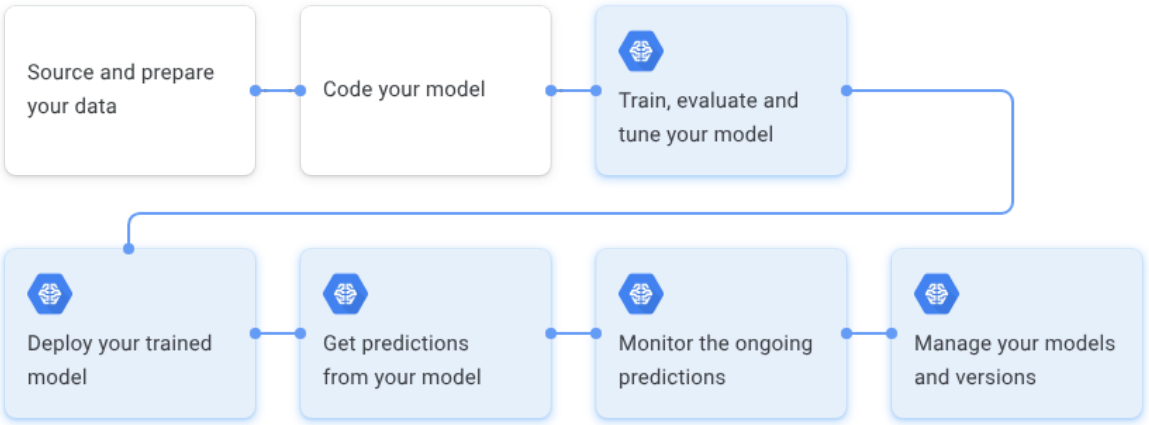


**Figure 2: The ML Project Lifecycle**

Today Machine Learning systems are able to tackle a plethora of problems, such as Image Recognition, Object Detection, Named Entity Recognition, Clustering and much more [14] .

### 1.2.2 Types of Machine Learning Systems

Machine Learning Systems can be broken down into various categories, based on a number of criteria [6].

- **Supervised learning:** This approach entails the inclusion of the desired output in the training set, often called *labels*. A typical supervised learning task is *Classification* [13]. In classification tasks, the model is provided with the goal of mapping an input into a discrete (non-decimal) output, often called a class. A typical example would be sentiment classification, where given an example film review, the model should infer whether the review is *positive* or *negative* (each sentiment given by the numbers **0** and **1** respectively). This is an example of binary classification, but classification can also be multiclass, i.e. the number of classes is a result of the very nature of the task at hand. Another supervised learning task would be *Regression* [13], where the model attempts to predict a target continuous numeric value. Regression could be

used for price or stock prediction based on a set of features, e.g. the price of a house based on the number of bedrooms and its area in square meters. Common Classification algorithms include Logistic Regression, Naive Bayes and Support Vector Classifiers, while for Regression, Linear, Lasso and Ridge Regression are often used [6] .

- **Unsupervised Learning:** In unsupervised learning tasks, the input data is unlabeled [6] and the system attempts to infer without the desired outputs or any human intervention whatsoever. A common example of unsupervised learning is *clustering*. This task is an attempt to group similar data points, for example divide a company's customer base in groups based on age, residence or common interests. Tasks related to clustering are *Dimensionality Reduction*, which aims to simplify data without the excessive loss of information, as well as *Anomaly Detection*, often used for preventing fraudulent transactions: The system is given only the data points pertinent to normal transaction, and detects anomalies based on, for example, the amount of a potential fraudulent transaction. Common clustering algorithms include K-Means and DBSCAN [6], both of them based on data point density in a geometric space.

- **Semi-Supervised Learning:** Labeling data can prove to be a time and resource-consuming task, and one is often presented with datasets that are partly labeled. Face recognition can work in a semi-supervised manner. When given an amount of images of the same person, labeling the person in one image alone can potentially be enough for the algorithm to cluster it along with the rest of the (unlabeled) images of the same person. This means that both supervised and unsupervised methods can work in tandem [1].

- **Self-Supervised Learning:** Generating a labeled dataset from an unlabeled dataset is always a viable approach to any given task. Once labeled, a supervised learning approach can be implemented [6]. This approach is also used in image recognition, where each training image is partially masked and the model is trained to reconstruct the original image. The masked images are used as inputs, while the original images are used as labels. This kind of image reconstruction, when applied, say, to a dataset containing unlabeled pictures of pets, is able to lead to the development of a model that can discriminate between the various species.

- **Reinforcement Learning:** Approaches such as this [26], involve an *agent* that can observe its current environment, and perform various actions that can lead to *rewards* or *penalties*. The agent must become acquainted with the best strategy that is able to ensure the highest possible reward, ultimately called a *policy*. Reinforcement learning is often used in robotics in order to perform basic tasks (e.g. lift an item or walk) as well as in programs such as DeepMind's AlphaGo [6], which formed its policy by analyzing millions of Go games and playing against itself.

As far as *Classical Machine Learning* is concerned, i.e. algorithms associated with supervised or unsupervised learning, a major drawback can be observed, closely

associated with data preparation and processing. In order to maximize the inferential capacity of a given model, the training samples have to be transformed into meaningful sets of features (Recall that the price of a house can be calculated given its size etc). This process, called feature engineering [13], can prove to be both challenging and time consuming. Recent advancements in hardware, however, have presented a more viable solution: *Artificial Neural Networks*.

### 1.3 Deep Learning

The McCulloch-Pitts *artificial neuron,* a concept that has existed since 1943 [19] and inspired by neuroscience, could be considered as the first forerunner of the whole field of Deep Learning. Originally intended as an insight on the exact way of how biological neurons and synapses process information, by working in tandem to perform multiple computations based on propositional logic. However, given our current (yet very imperfect) understanding of the inner workings of the brain, one should consider biological and artificial neurons similar in name only. Nevertheless, the impact of the *artificial neuron* should by no means be underestimated, since in 1957 it gave rise to Frank Rosenblatt's *Perceptron* [19], it being one of the earliest examples of binary classification systems. Here, a slightly modified version of the artificial neuron is employed: The neuron applies a simple linear transformation to the numeric inputs, consisting of a dot product of a weight vector **w** with the inputs **x** and the addition of a bias b, like so (see eq. 1) [6] :

$$\mathbf{w}^{T} \cdot \mathbf{x}+b \qquad (1)$$

After the linear transformation, a *step function* is applied. A common step function is the *Heaviside step function*, setting a threshold for determining the decision boundaries concerning each of the 2 classes. The following is a high level layout of a perceptron:

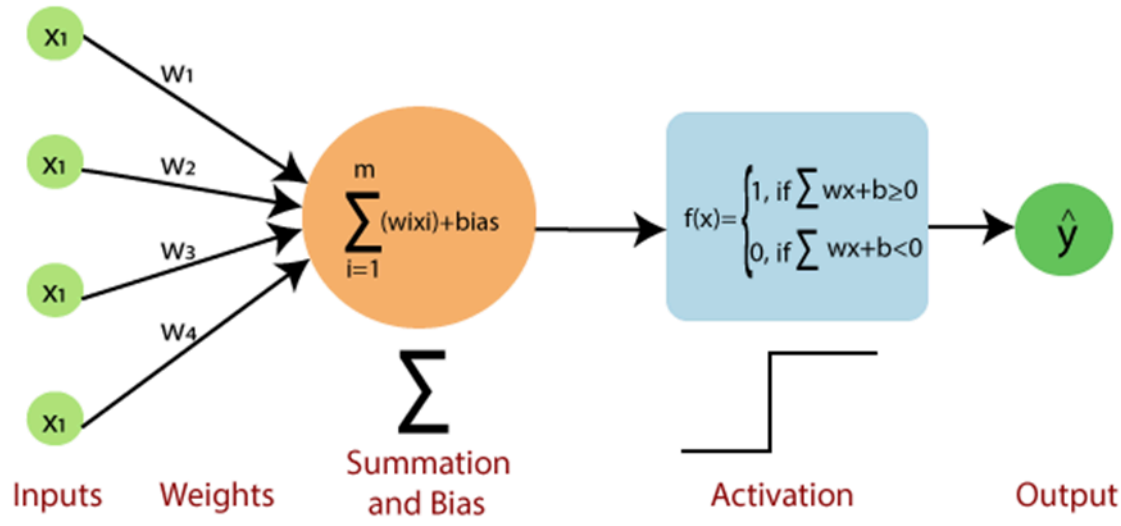


**Figure 3: The Perceptron**

Given that the Perceptron pertains to binary classification tasks, modern iterations have replaced the step function with the logistic (sigmoid) function [3], also used in

Logistic Regression. Generally, a perceptron can be organized by one or more artificial neurons, organized in a single layer, where every neuron (or node) is connected with every given input, such that the layers have taken the name *Dense* or *fully-connected* [7]. Modern neural architectures support the inclusion of multiple layers. Architectures such as these are often cited as *Multi-layer Perceptrons* or *Deep Neural Networks (DNNs)* [7]. Hence, the name Deep Learning stems purely from the layout of such architectures, consisting of an input layer, multiple hidden layers and an output layer. Computer Vision tasks are able to give a very helpful intuition behind the inner machinations of a Deep Neural Network [6]. The Network learns to discern smaller pieces of information when the input is propagated through the lower layers of the architecture (edges, corners) and slowly builds towards recognizing larger units (faces, buildings) in the final layers. This is how Deep Learning revolutionizes data processing, since feature extraction happens while information is propagated through the layers.

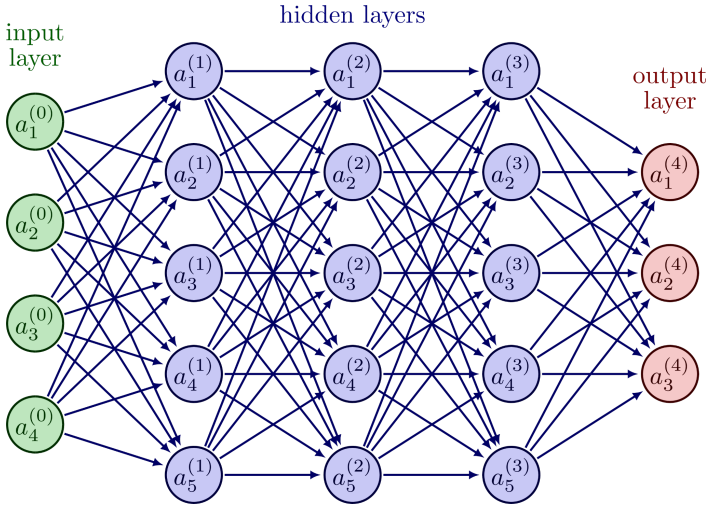


**Figure 4: Typical Feed-Forward Neural Network**

In order for a neural network to realize its full potential, a simple forward propagation will not suffice. Ever since the 1960s several researchers have entertained the idea of implementing some form of gradient-based optimization for MLPs. Seppo Linnainmaa's 1970 thesis [19] gave rise to the *reverse-mode automatic differentiation.* Through the implementation of this algorithm, the network gains the ability to perform gradient computation with respect to the model's parameters (weights and biases). The gradients can be used for performing a gradient descent step, which in itself can be used to describe how the network error varies when its parameters are tuned to a certain direction, thus providing an opportunity to adjust the coefficients in a way that minimizes the loss function. This combination of reverse-mode autodiff and gradient descent are implemented in an algorithm called *backpropagation* [13]. To summarize, predictions on the inputs as well as calculation of the total loss are made during the forward pass, while during the backward pass the algorithm measures the error contribution from each parameter and adjusts weights and biases to reduce the error (*gradient descent step*).

The implementation of the logistic function as opposed to the step function was done in the context of better accommodating the backpropagation algorithm in neural networks. The logistic function constantly has a well-defined non-zero derivative in contrast to the step function's flat segments. Other functions that work well in tandem with backprop are the *hyperbolic tangent* (*tanh*) and the *Rectified Linear Unit* (*ReLU*) [7]. Functions such as these are used as activation functions in the hidden layers of a neural network and are applied subsequently to the linear transformation occurring in each layer.

Neural Network architectures would eventually mature according to the needs and specifications of a given task. While *Convolutional* Neural Networks would later come to the aid of the booming field of Computer Vision, other tasks favored a more sequential way as far as processing information is concerned.

### 1.4 Sequence Models: Recurrent Neural Networks

*Recurrent Neural Networks* (RNNs) is a broad category of neural networks that is able to analyze time series data, e.g. the hourly temperature in a certain area. RNNs can work on sequences of arbitrary lengths, while ANNs and DNNs work with samples of fixed input. Said sequences are processed by way of iteration through the sequence elements as well as maintaining a *state* containing insights relative to already processed information, i.e. RNN nodes are built on an internal loop. An RNN comprising a single node receives inputs, processes them and passes the inputs back to itself. More specifically, at each timestep *t*, the neuron receives the input $\mathbf{x}_{(t)}$, while also receiving its own predictions from the previous timestep, $\hat{y}(t\text{-}1)$. This process is called unrolling the network through time. Every neuron has two sets of weights: one for the input $\mathbf{x}_{(t)}$, and another for the predictions of the previous timestep $\hat{y}(t\text{-}1)$ as well as a singular bias **b**. The weights, given with the notation $\mathbf{W}_x$ and $\mathbf{W}_{\hat{y}}$ are matrices, and the bias **b** is a vector [7].

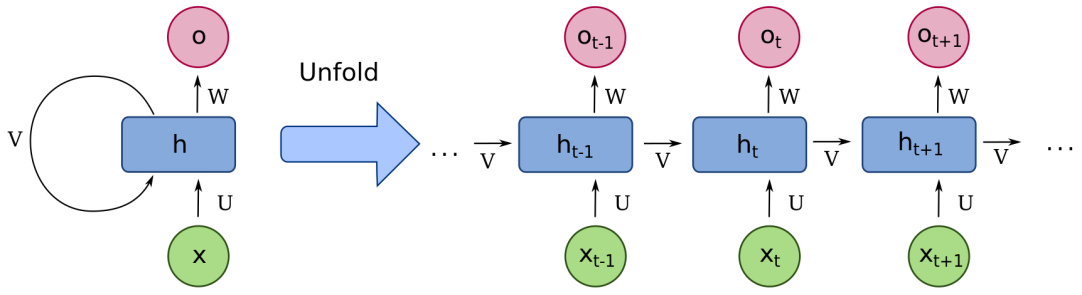


**Figure 5: Recurrent Neural Network**

RNNs typically receive a sequence of inputs and produce a sequence of outputs. Each individual cell state at time step *t* is a function of the timestep inputs, as well as the state at the previous timestep (see eq. 2) [13]:

$$\mathbf{h}_{(t)} = f(\mathbf{x}_{(t)}, \mathbf{h}_{(t-1)}) \qquad (2)$$

Training RNNs is done using a strategy called *backpropagation through time*, where the forward pass occurs within the unrolled network. What follows is the calculation of the loss function regarding all timesteps (see eq. 3):

$$\mathcal{L}(\mathbf{Y}_{(0)}, \mathbf{Y}_{(1)}, \ldots, \mathbf{Y}_{(t)}) \quad (3)$$

The gradients of the loss are then propagated backward through the unrolled network. At the end of that phase, an optimization step can be taken in a way identical to that of regular ANNs. However issues, such as *vanishing* and *exploding gradients*, arise during backpropagation, stemming from the gradients growing or diminishing excessively (below or above 0 respectively). Issues such as these are mitigated by implementing techniques such as *gradient clipping* [6].

### 1.4.1 RNN Variants

Transformations during data propagation through an RNN can lead to information loss between timesteps. Given an adequate amount of timesteps, information retention pertaining to the initial input state can be scarce or nonexistent. Architectures such as the LSTM (*Long-short Term Memory*) and the GRU (*Gated Recurrent Unit*) can handle such issues to the degree of substituting regular RNNs in a plethora of tasks [7].

Proposed in 1997 as a means of combating the issue of vanishing gradients [6] and gradually improved over the course of time, the LSTM is able to detect longer-term patterns in data, while leading to better convergence. Not quite dissimilar to a regular RNN cell, an LSTM cell's state is dependent on two different states, the *hidden state* $\mathbf{h}_{(t)}$ as well as the **cell state** $\mathbf{c}_{(t)}$. Both states are represented as vectors and are intuitively related to short term and long term memory respectively.

The LSTM cell is composed of a main layer that has the role of analyzing the inputs and the previous hidden state, much like a regular RNN. However, it also comprises three *gate controller* layers [13], whose activation or lack thereof is controlled by a Logistic Regression function (if the output is 0 every respective gate is closed, while if the output is 1 it remains open). The *forget gate* ($\mathbf{f}_{(t)}$) controls the parts to be erased from the long-term state $\mathbf{h}_{(t)}$, the *input gate* ($\mathbf{i}_{(t)}$) controls which parts of the main layer output should be added to the long-term state, and the *output gate* ($\mathbf{y}_{(t)}$), which controls the parts to be output at the current timestep both to $\mathbf{h}_{(t)}$ and $\mathbf{y}_{(t)}$.

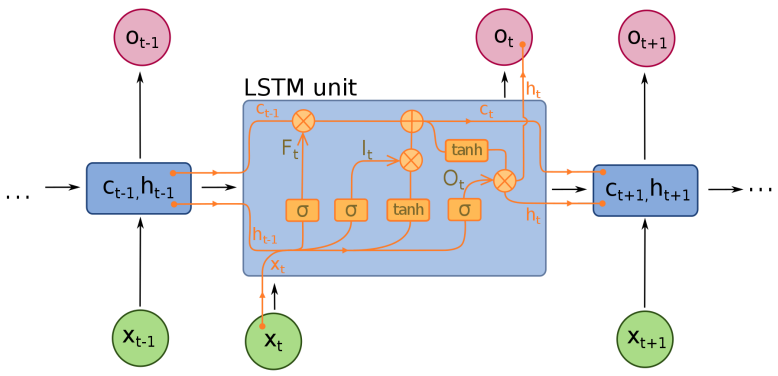


**Figure 6: Anatomy of a LSTM cell**

GRUs were introduced later in 2014 as a simplified version of the LSTM that displays quite similar performance to its more complicated counterpart. In a GRU cell, both state vectors are merged into a singular vector $\mathbf{h}_{(t)}$, while a singular gate controller $\mathbf{z}_{(t)}$ controls both the forget and input gate. The output gate is also absent, since the full state vector is the output of every timestep [7]. Even so, a new gate controller $\mathbf{r}_{(t)}$ controls the part of the previous state to be shown in the main layer.

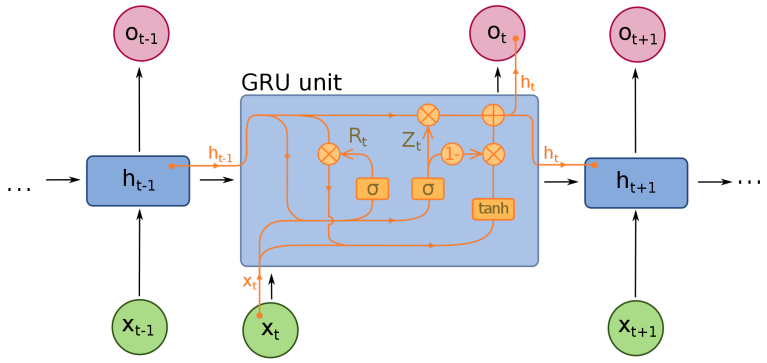


**Figure 7: Anatomy of a GRU cell**

Other techniques that have facilitated the improvement of RNN architecture are the *Bidirectional RNN*, that enables unrolling the RNN in both directions (forward and backwards in time), thus mitigating memory loss, and the encoder-decoder architecture composed of 2 RNNs working in tandem [13]. The former, namely the *Encoder* outputs a context vector, intuitively the gist of the inputs, which is propagated into the latter, the *Decoder* that yields the recurrent outputs. Most Encoder-Decoder architectures are further enhanced by the usage of an attention mechanism, essentially a layer that applies *attention weights* to the parts of the input that seem to be of the most significance [1].

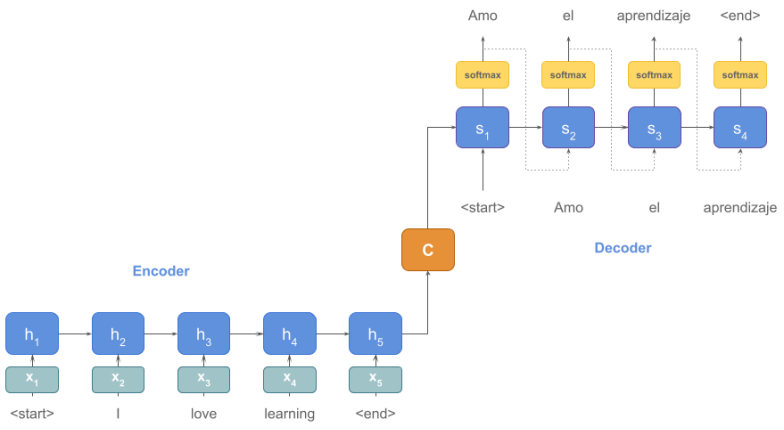


**Figure 8: Encoder-Decoder Architecture**

Encoder-Decoder architectures are one of the many factors that have recently revolutionized tasks such as machine translation as well as the field of *Natural Language Processing* in general.

## 1.5 Summary

Examining past efforts, one can only realize that precursors to the architectures used to tackle modern problems such as machine translation have been available for years. Machine Learning and in particular Neural Networks might have been a well established solution in tackling modern AI-related tasks, but it was only through high-quality datasets available over the itnernet, as well as rapid advancements in hardware that the Deep Learning Revolution could come into full fruition. The next chapter examines a rich and diverse field that has greatly benefitted from the Deep Learning boom: Natural Language Processing.

# 2. NLP AND TRANSFORMER ARCHITECTURES

The current chapter examines the evolution of the field of Natural Language Processing, a field of research that is not exclusively tied to, but has benefited a great deal from the field of Deep Learning. Great emphasis is given to concepts, such as Word Embeddings, and transformer architectures, such as BERT -currently considered as state-of-the art in natural language tasks- are introduced, specifically Greek-BERT, the DistilBERT architecture, as well as Knowledge Distillation that powers it.

## 2.1 Natural Language Processing

Even today, one of the most challenging tasks in the field of Artificial Intelligence is the understanding and generation of human natural language by computers. The main challenge of the field of Natural Language Processing (NLP) is that contrary to programming languages, human language rules fluctuate and evolve according to utility and usage. This kind of constant flux and evolution along with phenomena such as figurative speech, grammatical and syntactic ambiguity and regional variety, have given rise to a thriving interdisciplinary field, where computer science, linguistics and mathematics work in tandem to serve a common purpose. Given the abundance of text data over the internet, NLP might be more relevant than ever.

Symbolic AI approaches [6] to NLP were naturally the first step in creating systems facilitating *Natural Language Understanding* (NLU) and *Natural Language Generation* (NLG) [14]. Initial attempts to hard code the ruleset of a language in a programming language like LISP were met with partial success at best, while the first attempts in Machine Translation, a brainchild of the Cold War itself (Russian to English translations were thought to be a very valuable tool), were very limited both in performance as well as capability. The field of Applied Linguistics saw the collaboration of both engineers and linguists in creating the first simple chatbots: The ELIZA program, which came into fruition during the 1960s, utilized pattern matching to generate rather simplistic responses to textual prompts [19].

During the late 1980s and the 1990s, the availability of better hardware as well as the ubiquity of data started giving rise to more automated approaches [7]. Classical Machine Learning approaches seemed to be the go-to approach. Decision Trees seemed to be the natural next step to hard-coding and rules such as if statements, until more statistical methods like logistic regression came into the foreground [19]. Evolution was rather slow until the 2010s when rigorous feature engineering was abandoned in favor of Recurrent Neural Networks and in particular LSTM architectures.

Aside from other preprocessing steps, such as tokenization (converting the text into tokens, intuitively a list of words) or standardization (lowercasing, deaccenting, stemming, lemmatization). An important feature of NLP approaches is that due to the inability of machine learning algorithms to process raw text, said text has to be transformed into meaningful representations. Chunks of texts such as words, sentences, or paragraphs need to be vectorized. A popular method is that of word embeddings, implemented by algorithms such as GloVe, Word2Vec and FastText

[14]. Algorithms such as these enable the representation of a given word as a vector with certain dimensionality and are able to capture a significant portion of a word's semantic or syntactic capabilities.

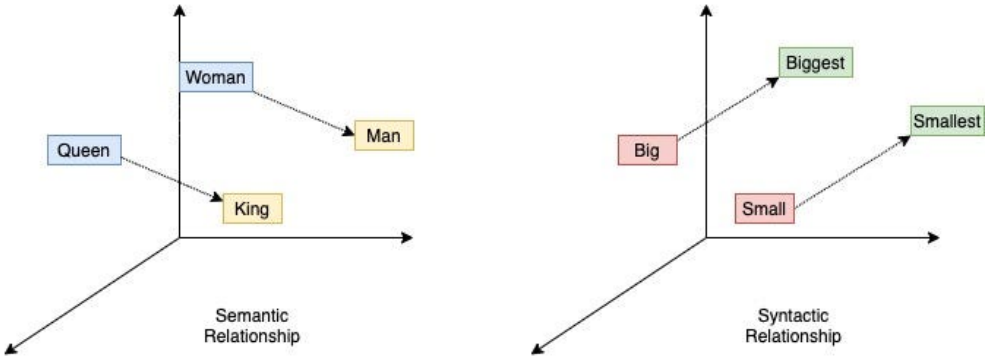


**Figure 9: Word Embeddings in a 2-dimensional space**

Dimensionality reduction in a 2-dimensional space [13], shows that the semantic difference between the vector representations of the words "woman" and "man" is analogous to that of "queen" and "king". The same goes for the syntactic difference between "big" and "biggest", and "small" and "smallest" [14]. What follows is a non-exhaustive list of common tasks involving NLP and word embeddings [6]:

- **Text Classification:** Assignment of a particular class to a given text. Instances of Text Classification come in the form of Sentiment Analysis (is a review positive or negative?), Content Filtering (Spam Detection, Abuse Detection).

- **Token Classification:** Assigning a class to every particular token in a given text. Notable examples include Named Entity Recognition (is a particular word a person, location or organization?) and Part-of-Speech Tagging (is a particular word a noun a verb or an adposition?).

- **Language Modeling:** Given a partly masked or incomplete sequence of tokens, predict the tokens that fit best in completing the sequence. Instances of Language Modeling include Masked Language Modeling and Text Generation as implemented by LLMs.

- **Speech Recognition and Speech-to-Text:** Given an audio file, yield the transcript of the text uttered within the duration of said audio in a written format or vice versa.

- **Automatic Summarization:** Given an input text, yield a summary containing only the most important chunks (sentences) with regard to the topic. Summarizations can be both extractive (keeping only certain sentences from the main text) and abstractive (distilling the basic idea of the text itself).

- **Question Answering:** Given an input natural language question, yield the correct answer. Question answering systems can be both extractive (keeping only the correct answer within a span of text) or graph-based (given a natural

language question, convert certain text spans into meaningful information within a certain graph, i.e. an entity and a relation).

- **Machine Translation:** Translate an input sequence from one language to another, e.g. from English to Spanish.

Tasks such as Machine Translation have proven to be a catalyst when it comes to the evolution of the field. From the first imperfect statistical endeavors, to the Encoder-Decoder architecture along with the implementation of the attention mechanism, Machine Translation has proven to be a quite challenging task, due to vast differences when it comes to grammatical and syntactical structures between languages. A giant leap forward would happen in 2017, when recurrence was completely abandoned and attention proved to be the most valuable component.

### 2.2 Transformers

In 2017 Vashwani et al. published the seminal paper *Attention is all You Need* [28]. The paper heralded the advent of the *Transformer* architecture that is the core of all current Large Language Model architectures. Built with the Machine Translation task in mind and similar to previous NMT architectures, the Transformer comprises an Encoder and a Decoder, but eschews recurrent layers in favor of multi-head attention. Hence all layers in the transformer are composed of numerous Multi-Head Attention Layers, with a Dense layer following in sequence after each attention layer. Multi-Head Attention is based on simple scaled dot-product attention, given by the following formula, intuitively mapping a query to sets of key-value pairs (see eq. 4):

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (4)$$

Where ***Q, K*** and ***V*** stand for *Query, Key* and *Value* respectively, each of them a matrix of weights with a dimension of *batch_size * sequence_length * embedding_dimension.* Multi-Head attention allows for the linear projection of the Queries, Keys and Values according to the number of the model's Attention Heads, i.e. a Transformer architecture with 3 attention heads calls for the initialization of three different query, key and value matrices per Multi-Head Attention layer. The dot product of the queries and keys is scaled by the square root of the dimensions of the keys matrix, run through a softmax function and finally multiplied by the value weight matrix [22].

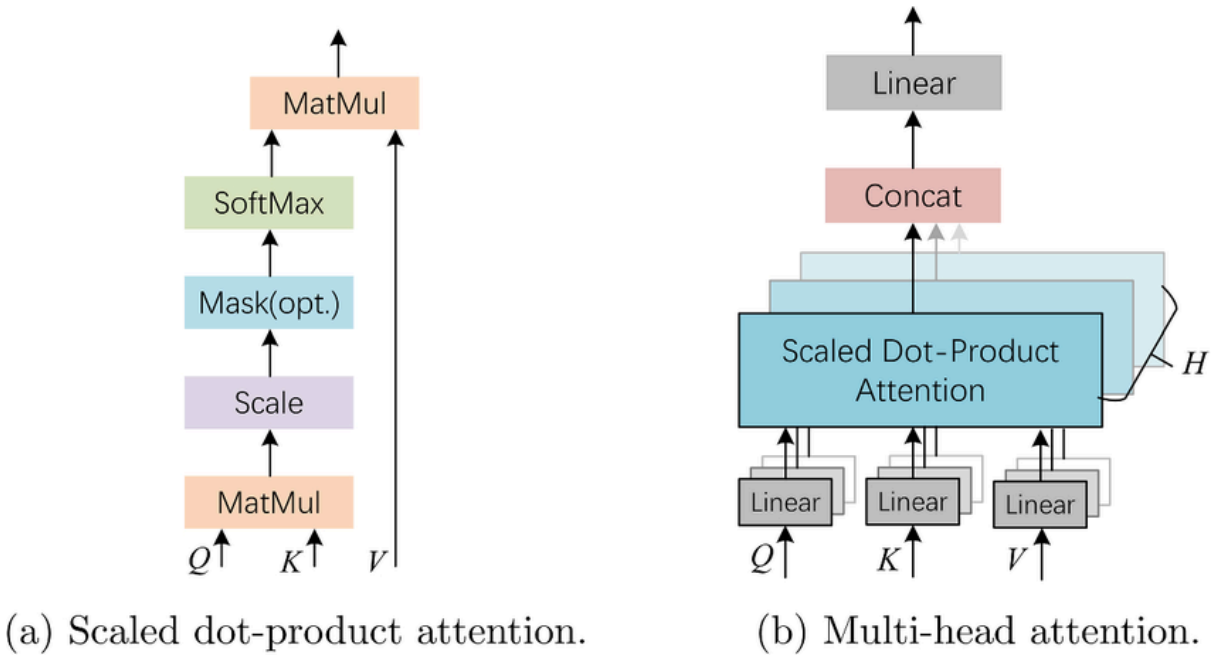


**Figure 10: Main mathematical operations in Transformers (from paper [28])**

Long range dependencies are captured by *Positional Encodings*, that represent the position of words within a sequence. Influenced by Digital Signal Processing and generated by using sine and cosine functions, Positional Encodings are dense vectors, much like word embeddings, and each positional encoding is summed with its corresponding word embedding, thus maintaining a semblance of word order even without recurrence [28].

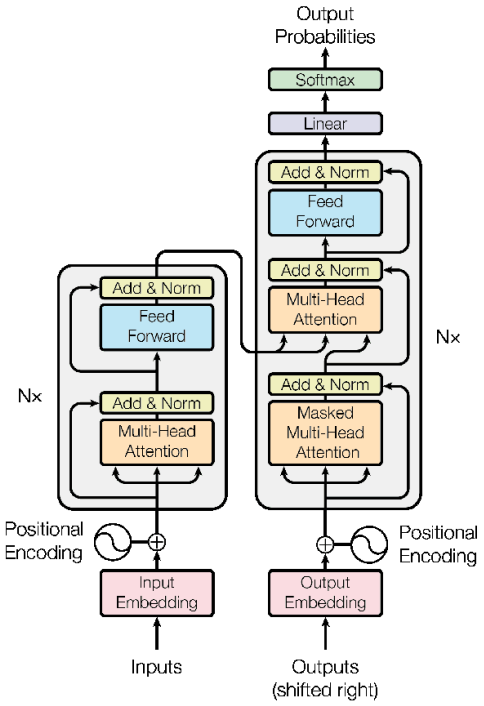


**Figure 11: Encoder-Decoder Transformer architecture (from paper [28])**

The original Transformer architecture was composed of 6 transformer blocks, each of them with an encoder as well as a decoder. The Encoder comprises a single attention layer followed by a linear network, both of them having a Normalization layer (LayerNorm) after them. When it comes to the Transformer Encoder, its Multi-Head Attention Layer performs an update on each word representation, with attention weights enriching the initially vague representations of words. The final output of the encoder is produced after traversing the FFN that is next in the sequence (composed of 2 dense layers, the former having a ReLU activation, the latter having no activation at all). followed by the LayerNorm. The input sequence $(x_1,...,x_n)$ - in machine translation that would be the sentence in the source language - is turned into meaningful representations $(z_1,..z_n)$. The output is then fed to the decoder's attention layers [22].

The decoder's lower attention layers perform the exact same function, but when processing each embedding it avoids processing the following ones, i.e. when processing the word "ate" in the sentence *the cat ate the mouse*, the layer focuses on "ate" and the words that come before it, a technique called *Masked Multi-Head Attention*. The upper decoder attention layer, along with the output of the previous layers receives the encoder output by means of a residual connection so as to apply *cross-attention*. Intuitively in the context of Machine Translation this means that during the generation of the words of the target sequence, the transformer attends to the words of the sequence in its source language as well as the words of the target sequence that have already been generated. When given two sentences "The cat ate the mouse" (source) and "Η γάτα έφαγε το ποντίκι" (target), when generating the word "ποντίκι", cross-attention pays as much attention to the word "το" as well as the word "mouse" [21].

Finally after traversing a final DNN followed by a LayerNorm, the output is run through a softmax activation and the final probabilities of the output sequence $(\hat{y}_1,...,\hat{y}_n)$ are returned, which in the task of machine translation would be the words of the target sequence. Aside from achieving state-of-the-art performance, transformer architectures enable better parallelization, while also reducing training time. Over time, the original transformer architecture would be refined and improved upon through Sequence-to-Sequence models like T5, Decoder models like GPT, as well as Encoder-only models like BERT [26].

### 2.3 BERT-based Architectures

Bidirectional Encoder Representations from Transformers (BERT) is a language model developed by Google, with the paper releasing in 2018 [4]. Like the name suggests, BERT is an encoder-only architecture, with the bidirectional element stemming from the fact that the encoder itself does not possess any Masked Multi-Head Attention Layers, a feature inherent in the original Transformer Decoder, as well as Decoder models, such as GPT [26]. BERT demonstrated the efficiency in pretraining a language model in a massive corpus, then using the pretrained model for inference on downstream tasks, such as text or token classification, in a process called *fine-tuning*, essentially an instance of *Transfer Learning* [7].

### 2.3.1 The Original BERT Model

The original BERT model was pretrained on BookCorpus (800M words) and Wikipedia (2,500M words). The process of pretraining entails two different tasks [4]:

**Masked Language Modeling (MLM):** Each word in every sentence of the corpus has a 0.15 probability of being masked. The model is trained to predict the masked tokens given an input sentence. For example, in a sentence such as *The* [MASK] *ate the mouse*, the model tries to complete the sentence by inferring that the masked word is *cat* [4].

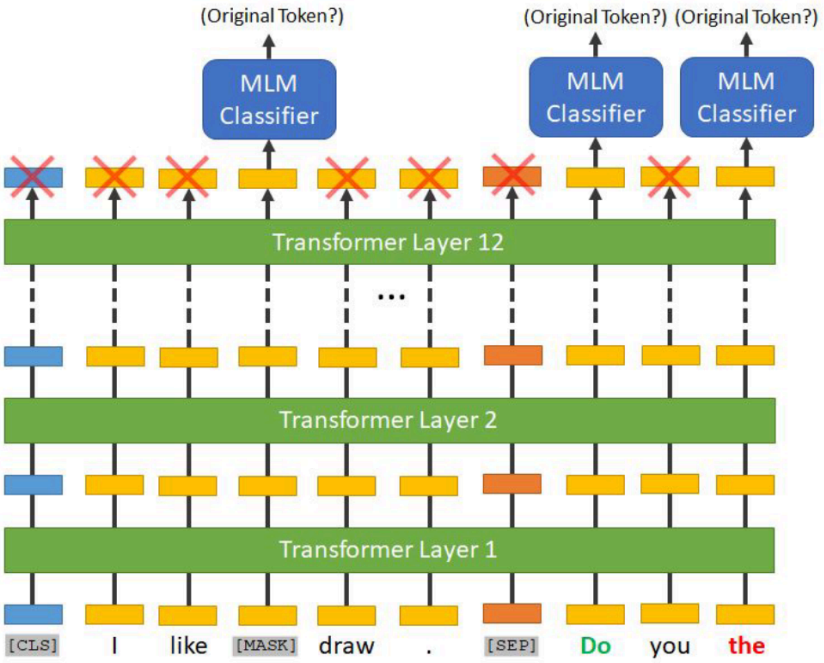


**Figure 12: Overview of the Masked Language Modeling Task**

**Next Sentence Prediction (NSP):** BERT is trained in predicting whether two sentences are actually consecutive in the corpus or not, essentially a binary classification problem. For example, given the sentences *the cat is sleeping* and *it's snoring loudly*, the model has to predict whether the sentences being consecutive is logical or not. More recent research has shown that NSP does not have the same importance as MLM during pretraining and has been rarely implemented in later architectures [17]. Even so, it seems to facilitate the model's inferential capability when fine-tuning on tasks such Natural Language Inference, where inter-sentential dependencies seem to matter the most [4].

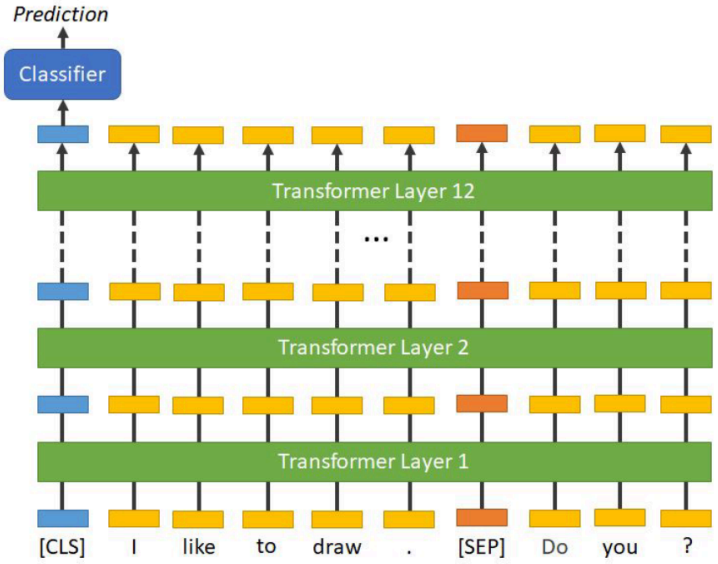


**Figure 13: Overview of the Next Sentence Prediction Task**

The model is trained simultaneously on both MLM and NSP. Specifically for the NSP task, as seen in the figure above, a [CLS] token is inserted at the start of every input and the corresponding output, propagated by a linear binary classification Feed-Forward Network, often called a *classification head* is the prediction, i.e. where the sequence of sentences is logical or not, them separated by the special [SEP] token. In the case of MLM, the loss is computed only on the masked tokens, and in the case of NSP, only on the prediction of the classification head. After pretraining, the model is fine-tuned on different tasks, retaining the pretraining weights, with the only modification being a brand new classification head [4].

BERT comes in a variety of architectures, such as BERT-Base and BERT-Large, composed of 12 and 24 stacked encoder layers respectively. The base model has a hidden dimensionality of 724, while the large model has a hidden dimensionality of 1024. They also come in cased and uncased versions, i.e. taking into account or ignoring letter casing. BERT has a vocabulary of 30,000 tokens, created via the implementation of the WordPiece tokenization algorithm as a means to handle OOV (out-of-vocabulary) words. What this means, essentially, is that if the BERT Tokenizer does not recognize a word, it breaks it down into smaller sub-word units [4, 14].

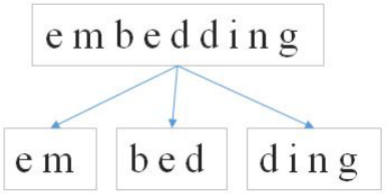


**Figure 14: Example of WordPiece Tokenization**

Being an encoder model, BERT has an affinity in classification and extractive question answering tasks, but lacks the capabilities of decoder model in Sequence-to-Sequence tasks, such as summarization or machine translation, or even text generation which was greatly refined by the GPT architectures [26]. Even so, it sparked an avalanche of transformer encoder models greatly influenced by it. While some of them were focused on robust pretraining and optimization, some of them were more in tune with the linguistic capabilities hidden in the pretraining process.

### 2.3.2 GREEK-BERT

Language-specific iterations of BERT had to be developed in order to mitigate the fact that the original BERT model was pretrained in an exclusively English language corpus. Multilingual models such as M-BERT were trained on multiple language iterations of the Wikipedia corpus (104 languages) [26], but given the fact that such an attempt is prone to language-specific biases, due to the distribution of the availability of Wikipedia texts in different languages itself being highly uneven, monolingual models were the next logical step [18].

Like CamemBERT for French or GottBERT for German, GREEK-BERT was a proposed answer to the necessity of a language model designed specifically for modern Greek. Released in 2018 by a team of researchers affiliated with the Athens University of Economics and Business, GREEK-BERT was based on the BERT-base-uncased architecture, having 12 stacked encoder layers and attention heads, and a hidden size of 768. The vocabulary size is slightly increased, it being at 35,000 tokens, generated by training a SentencePiece tokenizer. Pretraining came down to the tasks of MLM and NSP, with the pretraining corpus having a size of approximately 29GB. The corpus consisted of the Greek part of WIkipedia (730 MB in size, consisting of approximately 0.08B tokens), the Greek translation of the European Parliamentary Proceedings Corpus (Europarl, it being around 380MB, and 0.04B tokens long), and finally the majority of the corpus was made up of OSCAR, a relatively cleaned and preprocessed version of the Common Crawl, consisting of articles scraped directly from web pages (27GB, 2.92B tokens) [16].

What follows is the evaluation and benchmarking in 3 downstream tasks on smaller Modern Greek corpora: Part-of-Speech Tagging, Named Entity Recognition and Natural Language Inference. GREEK-BERT outperformed previous architectures in all tasks but POS-tagging in spite of attaining an accuracy of over 98%. NER accuracy levels ranged from 84.7 to 86.7%, as well as approximately 78-79% for NLI.

The GREEK-BERT model was one of the main influences in designing the architecture later proposed in the context of the present thesis. However, when it comes to identifying its key component, one has to examine a very different implementation of the BERT architecture.

### 2.3.3 DistilBERT and Knowledge Distillation

BERT-based architectures have a significant downside pertinent to their sheer size in parameters. Overly long training times as well as constantly rising demands in computational power, have rendered large models such as these not always viable for users outside the boundaries of well-funded research projects or large corporations. Recall that the original transformer architecture was trained on 8 P100 GPUS, while the base BERT model was pretrained on 4 Google Cloud TPUs (16 TPU chips in total). The cost of such processes can rise to hundreds of thousands of dollars and the subsequent energy consumption is very detrimental to the environment to say the least [26].

For such reasons, attempts have been made in democratizing and downsizing language models, as well as improving data-driven efficiency. Towards the end of 2019, Victor Sanh and a team of researchers associated with the Hugging Face platform released DistilBERT, a smaller, cheaper alternative to the BERT architecture that is able to retain a staggering amount of the original model's performance capabilities. More specifically, DistilBERT is able to retain 97% of the original model's performance, while being 60% faster and reducing the number of parameters by 40% (66M parameters as opposed to BERT's 110M) [23].

That level of compression is owed to a technique named *Knowledge Distillation*, where a student model (here DistilBERT) is trained in reproducing the behavior of a larger counterpart, called the teacher (in our case the original BERT model). The main idea stems from the fact that during supervised learning, a standard objective is the minimization of the cross-entropy between a model's predicted labels and the one-hot empirical distribution of said labels, i.e. softmax probabilities on incorrect classes should have a tendency towards 0. Knowledge Distillation comes [10] in a variety of implementations showing differences in the usage of a specific distillation algorithm, as well as loss calculation and teacher-student architecture. Popular types of Knowledge Distillation (KD) include [10, 26]:

- **Response-based Distillation:** The student is trained to mimic the predictions of the teacher model by minimizing the difference between the predicted outputs. During the distillation process, the teacher generates soft labels, e.g. softmax probabilities over specific classes. The student model is trained to predict the same soft labels as the teacher model by minimizing a loss function between their predicted outputs.

- **Feature-based Distillation:** The teacher model is first trained on the training data to learn all relevant task-specific features. The student is then trained to learn the same features by minimizing the distance between the features learned by student and teacher respectively. Teacher representations are extracted from an intermediate layer and used as targets for the model. Popular loss functions include the Mean Squared Error (MSE) as well as the Kullback-Leibler Divergence (KLD) [26].

- **Relation-based Distillation:** The teacher model generates a set of relationship tensors that capture dependencies between inputs and outputs. The student is then trained in learning the relationship tensors by minimizing a

loss function that measures the difference between the relationship tensors between teacher and student.

- **Adversarial Distillation:** Commonly used in Generative Adversarial Networks (GANs). The adversarial (student) network is trained to generate training samples that might prove difficult for the teacher to classify correctly. Synthetic data such as this is generated by the addition of perturbations to the original training samples. The student is then trained to classify both the original and synthetic data.

Other KD types include Offline and Online Distillation, Multi-Teacher and Cross-Modal Distillation techniques. Below we can see the type of KD applied specifically during the pretraining phase of the DistilBERT model.

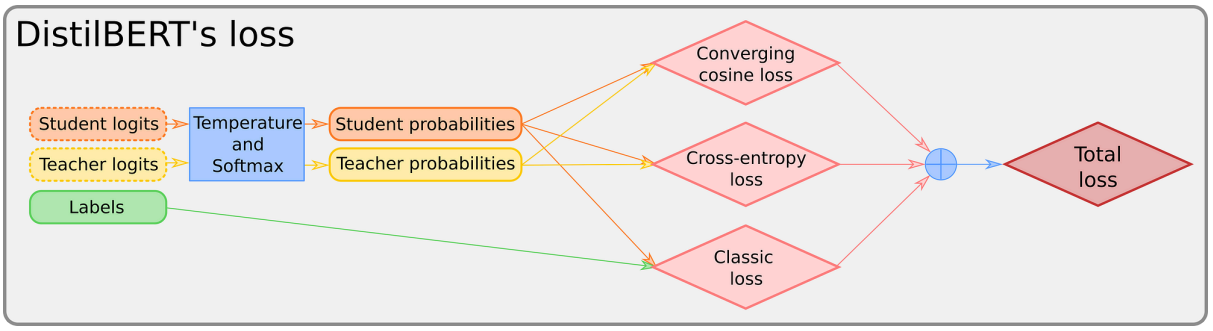


**Figure 15: Calculating DistilBERT's pretraining loss**

Pretraining consists of a singular task, i.e. Masked Language Modeling (MLM) on the Toronto Book Corpus [23]. Student initialization was made possible due to the common dimensionality between teacher and student. The student model's performance is measured using a *distillation loss* over the soft target probabilities of the teacher, i.e. the model essentially measures the difference between the softmax probabilities calculated by teacher and student respectively (see eq. 5):

$$L_{cross} = -\sum_{i=1}^{n} t_i * \log(s_i) \tag{5}$$

Where $t_i$ and $s_i$ refer to the probability outputs of the teacher and student models respectively. Influenced by Geoff Hinton's insights on Knowledge Distillation for the field of computer vision a *softmax-temperature* was also implemented for calculating the soft targets of a vector **z** hard target:

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{6}$$

Where temperature *T* controls the smoothness of the output distribution and $\mathbf{z}_i$ is the model score for every class i (see eq. 6). The temperature is applied to both student and teacher, while at inference, *T* is set to 1 to recover a standard softmax. The final

training loss is calculated by a linear combination of the distillation loss $L_{ce}$ as well as the pretraining loss on the MLM task itself, $L_{MLM}$ and finally adding a *cosine embedding loss* ($L_{cos}$) in order to align the direction of the hidden states vectors between student and teacher [23].
As mentioned above, benchmarks on datasets such as General Language Understanding (GLUE), as well as the Stanford Question Answering Dataset (SQUAD), show that the DistilBERT model retains 97% of the original BERT model's accuracy. Models such as these, where lower computational necessities and smaller training times come at the cost of sacrificing a meager amount of performance are perfect candidates for domain adaptation. What follows is a rigorous description of the design, implementation and evaluation of a similar architecture developed specifically for the Greek language.

## 2.4 Summary

Transformer-based architectures have been solidified as the dominant option in NLP-related tasks, due to their significant accuracy and speed, when compared to previous RNN architectures. Furthermore, techniques, such as Knowledge Distillation appear to be a viable form of compression, that decreases hardware requirements, increases speed and keeps a great deal of a decompressed model's accuracy intact. This advantage will be leveraged when training a distilled transformer encoder model for the Greek language.

# 3. DISTILBERT-EL-KLD

Attempts have been made to develop Language Models for the Greek language. One such model is GREEK-BERT, used for a plethora of NLU tasks. Here, the original GREEK-BERT architecture is used as a basis for the development of a DistilBERT model specialized in Greek.

## 3.1 Research and Development

The model presented in the current thesis, namely DistilBERT-EL-KLD, is an attempt in creating an end-to-end infrastructure, starting from the very selection of the training corpora, to pretraining via a popular Knowledge Distillation method, and finally the evaluation and benchmarking on the same corpora as GREEK-BERT.

### 3.1.1 Motivation

Even though architectures attempting to expand upon the original iteration of GREEK-BERT, have already been implemented, some of them being DistilBERT architectures, like the *distilbert-base-el-cased* architecture [11], developed among a plethora of multilingual distilled versions of BERT, this attempt differs from many of them in implementing the Kullback-Leibler Divergence (KLD) loss [26], contrary to the original distillation loss used by Sanh et al. [23] as well as using only a fraction of the original GREEK-BERT training corpus as measured in sheer size.

### 3.1.2 Tools and Resources

Pretraining was performed on an HPC node provided by the Cyprus Institute's preparatory access program[1], according to which, researchers can obtain completely free access on a high performance computer for tasks such as scalability testing and model training, for a finite amount of CPU and GPU hours. For the purposes of the current research, we utilized a node with 4 A100 GPUs, while being given 1000 GPU hours for pretraining purposes. For the downstream tasks, the T4 virtual GPU (16 GB) provided by Google Colab notebooks proved to be more than sufficient for the tasks, and fine-tuning relied solely on it.

Aside from a few bash scripts made specifically to schedule pretraining-associated tasks within the linux server, model design, training and evaluation was done by utilizing the Python programming language. Python's higher level APIs, such as PyTorch and Tensorflow, as well as a very active community make it a perfect candidate for Data Science and Artificial Intelligence research projects. The undertaking was carried through the usage of the PyTorch deep learning framework, developed by Meta AI, and was greatly facilitated by Hugging Face's Transformers, Datasets and Accelerate libraries [10]. The Hugging Face community has given way to the democratization of AI tools and methods by enabling users and researchers alike to implement deep learning models in a few lines of code, as well as providing memory-efficient ways in handling large corpora in addition to simplifying parallelization and training on multiple devices. The Pandas API was also used for

[1] https://castorc.cyi.ac.cy/access-to-resources/preparatory-access

data exploration, preprocessing and analysis, while graphs used in the section were generated by Matplotlib.

### 3.1.3 Pretraining Corpora

Datasets used for pretraining were essentially more recent versions (as of April 2023) of the ones used for pretraining the original GREEK-BERT architectures [16]. The initial corpus comprises the Greek section of Wikipedia [29], the Greek translation of Europarl [15], as well as the Greek rendition of OSCAR [24], a normalized (to some degree) iteration of the Common crawl corpus. However, only a small fraction of the given corpus was utilized during pretraining. After data selection and preprocessing, we were left with Wikipedia (464.1MB), Europarl (370.8MB) and the first 8 shards of OSCAR out of the original 70 (3.78GB, each individual shard averaging at around 480MB - the original size of the corpus in its entirety was 78.3GB, with over 5B tokens). Note that the sizes refer to the file after rigorous preprocessing. Especially the OSCAR shards were vastly downsized due to them being web-scraped content that contained a lot of noise text.

### 3.1.4 Preprocessing and Tokenization

Generally, the process consists of a universal pipeline with slight corpus-specific variations. Cleanup techniques are both intersentential, as well as intrasentential, with a mixture of regex-based substitutions and statistical-based sentence removal methods being applied. For correctly parsing the input text into sentences, a modification had to be made on the base NLTK sentence tokenizer, so as to accommodate symbols such as *;* and · (greek ano teleia) considered to be strong punctuation markers and thus sentence delimiters in the Greek language. The tokenizer was also modified so as not to split sentences in abbreviations, such as “π.χ.” or “κ.ο.κ”, that contain symbols that could erroneously be considered strong punctuation themselves. The initial text was broken down into articles, for Wikipedia and OSCAR (the former’s articles being separated by HTML <doc></doc> tags and the latter being available in a JSON format, with each article stored within a “Content” key), while the Europarl corpus was already broken down into sentences, separated by a newline character. General preprocessing steps include the following:

- Removal of accents
- Lowercasing
- Removal of HTML tags, especially on OSCAR and Wikipedia
- Removal of hyperlinks (indicated by the https:// pattern)
- Removal of emails and internet usernames (indicated by the @ character)
- Removal of parentheses
- Removal of trailing spaces left and right of each text chunk

- Removal of spaces before punctuation characters
- Deduplication of consecutive punctuation characters, newlines and spaces
- Initial removal of all newline characters
- Removal of articles/Europarl sentences that have a very low count of Greek alphabetic characters. Specifically, a textual chunk is removed if 50% or less of its alphabetic characters are Greek.
- Removal of sentences that have a very low count of alphabetic characters. Specifically, if less than 50% of a sentence's tokens is completely composed of alphabetic characters, the sentence is removed.
- Removal of all overly short sentences, i.e. those that contain 5 tokens or less: in Wikipedia, those could be articles consisting only of the title, or disambiguation pages.
- Regarding the OSCAR corpus, sentences with certain keywords were removed: words such "σχόλια", "δημοσιεύτηκε", "προβολές" and "e-mail" often belong to textual chunks associated with a website's metadata, advertisements, contact information or comment sections, and thus considered too noisy to be included in the training dataset.
- Emojis and non-ASCII characters were also removed with the help of the original GreekBERT's normalization helper functions.

The NLTK sentence tokenizer outputs lists whose elements are an input text's individual sentences. In between those sentences, the newline character was added, so as to feed the pretraining model architecture with small logical segments of texts that were mostly devoid of random noise. The corpus shards were saved in .txt format. For tokenization purposes during pretraining, the original GREEK-BERT vocabulary was used [16], it being composed of 35,000 BPEs extracted by means of the sentencepiece tokenization library.

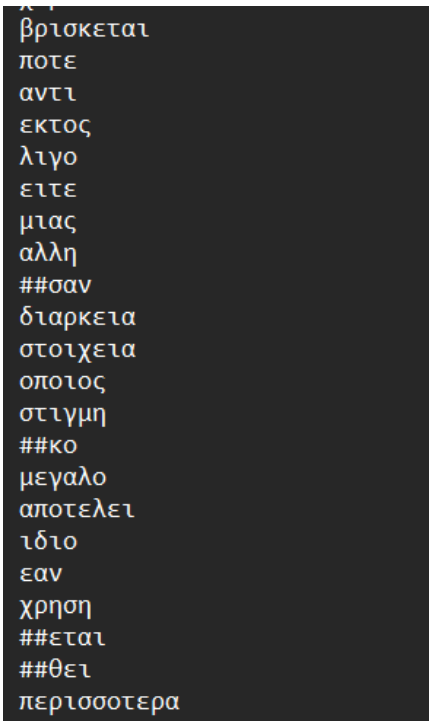


**Figure 16: Sample of the GREEK-BERT Tokenizer vocabulary file**

The vocabulary is composed of individual words in their entirety as well as sub-word segments as a means of handling OOV word instances. The completion of the preprocessing pipeline signals the start of the most computationally intensive task required in developing a model, that being pretraining.

### 3.1.5 Pretraining

Pretraining was done exclusively on the Masked Language Modeling task, following in the footsteps of the original DistilBERT paper. The original GREEK-BERT model was used as the teacher, and the exact same vocabulary was used for encoding and tokenization [16]. Wikipedia, Europarl and the seven first parts of the OSCAR corpus were used as a training set, and the eight OSCAR shards alone as a validation set. Due to differences in the training data being used, it was deemed necessary for the GREEK-BERT model to be fine-tuned on the masked language modeling task. The number of sentences was approximately 9M.

GREEK-BERT was pretrained for 80000 steps, with a masking probability set at 0.15, much like the original BERT. Sequences were padded and truncated to a token length of 512. Training was conducted by utilizing the Hugging Face Trainer API [10]. The training arguments are listed below.

**Table 1: Pretraining Arguments for GREEK-BERT**

| **Argument** | **Value** |
|---|---|
| per_device_train_batch_size | 16 |
| per_device_eval_batch_size | 16 |
| learning_rate | 5e-5 |
| weight_decay | 0.01 |
| evaluation_strategy | “steps” |
| max_steps | 80000 |
| eval_steps | 5000 |

GREEK-BERT ran for 80000 training steps over the course of approximately 36 hours. Evaluation was performed every 5000 training steps on a small subset of the corpus, namely 5000 training samples. Taking into account gradient accumulation and multiple GPUs, an effective batch size of 256 was achieved. Initial training loss (step 5000) was 1.95, while being 1.79 at the end of pretraining. Validation loss was 1.86 and 1.69 during the start and the end of the pretraining process respectively. Convergence was fast, since the updated corpus is not extremely different to that of its counterpart used by the original GREEK-BERT model. The final loss values are not that impressive, but would prove more than sufficient in order to get satisfying results on the downstream tasks.

Next, pretraining was performed using knowledge distillation. One significant modification is that the knowledge distillation loss function implemented is the Kullback-Leibler divergence [26], efficient in measuring the distance between 2 distributions, thus giving insight to the difference between the softmax probability outputs of teacher and student respectively (see eq. 7 below).

$$D_{KL}(p||q) = \sum_{i=1}^{N} p(x_i) log(\frac{p(x_i)}{q(x_i)}) \qquad (7)$$

$y_{true}$ being the soft probabilities of the student and $y_{pred}$ those of the teacher. Each of the distributions were scaled by a factor *T* (temperature), set to 2 during pretraining. Loss output is finally scaled by the square of the temperature in order to soften the distributions even further. The final loss is a linear combination between the KLD loss output and the actual cross entropy loss output by the student model modified by weights *alpha*. The final loss function was as follows (see eq. 8:

$$Loss = a * Loss_{ce} + (1 - a) * Loss_{KD} \qquad (8)$$

During pretraining, alpha was set to 1 so as to result in an unweighted loss function. Cosine embedding loss was not utilized since it proved to be inconsequential during the initial phases of pretraining [26].

Training lasted about 33000 steps, a little more than 2.5 iterations of the whole corpus. The process lasted about 10 days and 15 hours, with the trainer performing approximately 35,000 steps every 24 hours. Pretraining arguments were set accordingly.

**Table 2: Pretraining Arguments for DistilBERT-EL-KLD**

| **Argument** | **Value** |
|---|---|
| alpha | 0.5 |
| temperature | 2 |
| per_device_train_batch_size | 16 |
| per_device_eval_batch_size | 16 |
| gradient_accumulation_steps | 2 |
| evaluation_strategy | “steps” |
| learning_rate | 1e-4 |
| weight_decay | 1e-5 |
| max_steps | 330000 |
| eval_steps | 5000 |

Training time was considerably longer, and thus benefitted from a more aggressive, higher learning rate, as well as more conservative, lower weight decay rate. By default, the AdamW optimizer was utilized for optimization purposes. With gradient accumulation and multiple GPUs considered, an effective batch size of 128 was reached. The final checkpoint of the fine-tuned GreekBERT model was used as an initialization point for the teacher model, and the distilbert-base-uncased model from Hugging Face [10] was used as the initialization of the student. Initial training loss was at 5.67, while initial validation loss was at 4.51. During the final training step, training loss was at 2.12, and validation loss at 2.03.

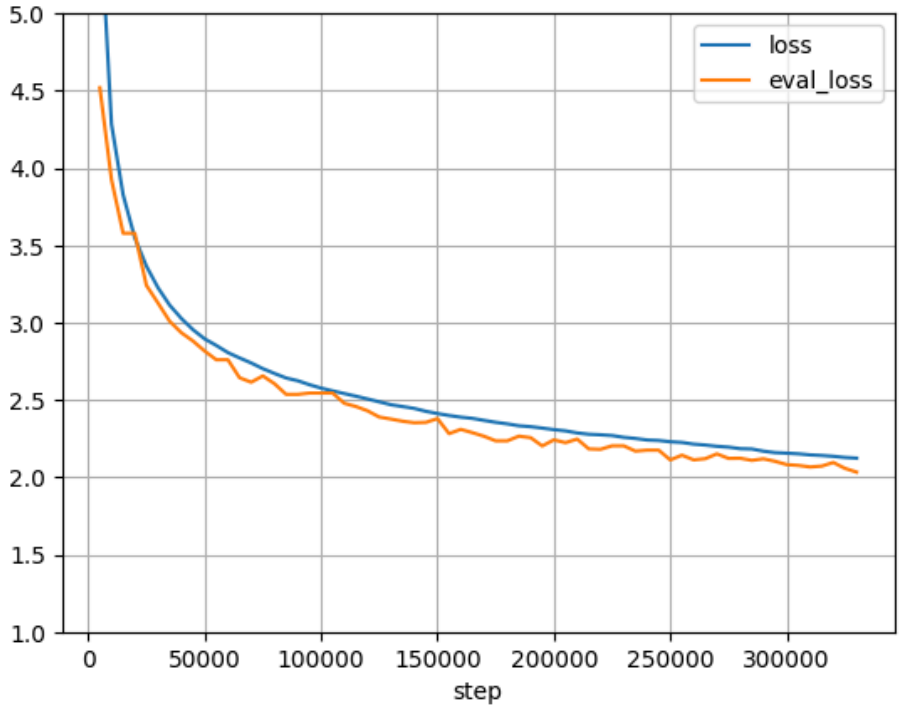


**Figure 17: Learning Curve for DistilBERT-EL-KLD**

One must keep in mind that results are far from ideal, since when given ample training time and resources, there is more room for implementing different parameters, redesigning training pipelines, or simply training for a longer period of time in order to minimize the loss function even further. Nevertheless, as we will see further, the current pretraining method was enough to make our model a very viable, low-cost alternative to the original GREEK-BERT when fine-tuning in downstream tasks.

### 3.2 Fine-Tuning on Downstream NLP Tasks

The tasks chosen for evaluating the pretrained model are identical to those used in evaluating the original GREEK-BERT architecture. The three tasks, namely Part-of-Speech (PoS) tagging, Named Entity Recognition (NER), as well as Natural Language Inference, were performed on smaller Greek corpora that were themselves built from raw input data, preprocessed and cleaned.

#### 3.2.1 General Assumptions

Evaluation was performed by using the poseval and seqeval metrics for POS-tagging and NER respectively [10], while results on the NLI task were given by Scikit-Learn's classification_report function. Metrics include precision, recall and F1-Score (it being the harmonic mean between precision and recall), as well as overall accuracy:

In all cases, both class-specific and overall scores were acquired and will be displayed along with those of the original GREEK-BERT in order to compare them and draw conclusions. Generally, the DistilBERT-EL-KLD proved to be very close to the original GREEK-BERT's performance benchmark, except maybe for NLI where the gap between the 2 different models starts to widen. Even so, the DistilBERT-EL-KLD proved to be by far the fastest option of the 2, and certainly the least computationally expensive. For fine-tuning purposes the Google Colaboratory virtual environment was used, as well as the T4 GPU that is provided for free.

### 3.2.2 POS-Tagging

Part-of-Speech tagging was conducted using the Greek Universal Dependencies Treebank (GUDT) corpus, developed by the Institute of Language and Speech processing of the "Athena" research center [20]. The dataset is split in *train*, *development* and *test* sets, containing 1,622, 403 and 456 sentences each. Each sentence has been annotated by tags corresponding to individual tokens. Each of these tags represents a token's part of speech, e.g. verb, noun etc. Along with the POS-tagging sections, the dataset contains syntactic annotation, that we've elected to ignore. The dataset files come in CONLL-U format. Each individual file comprises a number of sentences, denoted by a *sent_id*.

```
# newdoc id = gdt-20120309-elwikinews-5160
# sent_id = gdt-20120309-elwikinews-5160-1
# text = Η Μάντσεστερ Γιουνάιτεντ ηττήθηκε από την Ατλέτικο Μπιλμπάο με σκορ 2:3
1       Η       ο       DET     DET     Case=Nom|Definite=Def|Gender=Fem|Number=Sing|PronType=Art       2       det     _       _
2       Μάντσεστερ      Μάντσεστερ      X       X       Foreign=Yes     4       nsubj   _       _
3       Γιουνάιτεντ     Γιουνάιτεντ     X       X       Foreign=Yes     2       flat    _       _
4       ηττήθηκε        ηττώμαι VERB    VERB    Aspect=Perf|Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Pass    0       root    _       _
5       από     από     ADP     ADP     _       7       case    _       _
6       την     ο       DET     DET     Case=Acc|Definite=Def|Gender=Fem|Number=Sing|PronType=Art       7       det     _       _
7       Ατλέτικο        Ατλέτικο        X       X       Foreign=Yes     4       obl:agent       _       _
8       Μπιλμπάο        Μπιλμπάο        X       X       Foreign=Yes     7       flat    _       _
9       με      με      ADP     ADP     _       10      case    _       _
10      σκορ    σκορ    X       X       Foreign=Yes     4       obl     _       _
11      2:3     2:3     NUM     NUM     NumType=Card    10      nmod    _       _
```

**Figure 18: Example of the GUDT Corpus**

Exactly below the ID, is the full text. Instead of the full text, each individual token could be extracted, located in the second column right below it, next to their respective numbers. Columns are separated by tab (\t) characters and two columns to the left, there is a list of PoS tags, corresponding to each individual token. Following is a list of all POS tags used in the dataset.

**Table 3: List of Labels in the GUDT Corpus**

| | |
|---|---|
| **ADJ** | Adjectives: καλός, κακός, όμορφος, άσχημος |
| **ADP** | Adpositions, i.e. prepositions: αντί, κατά, προς |
| **ADV** | Adverbs: έτσι, ομοίως, πριν |
| **AUX** | Auxiliary Verbs: έχω (as in *έχω έρθει*) |
| **CCONJ** | Coordinating conjunction: και, παρά (as in *καλύτερα εγώ **παρά** εσύ*) |
| **DET** | Determiners, i.e articles: ο, η, το |
| **NOUN** | Nouns: μαθητής, στρατιώτης, γιατρός |
| **NUM** | Numerals: ένα, δύο |
| **PART** | Particles: όχι, δεν, μην |
| **PRON** | Pronouns: αυτός, εκείνος, οποίος |
| **PUNCT** | Punctuation: .,!; |
| **SCONJ** | Subjunctive Conjunction: ότι, όταν (as in *μου είπε **ότι** θα φύγει*) |
| **VERB** | Verbs: κάνω, τρέχω παίζω |
| **X** | Foreign words: Μάντσεστερ, Λιγκ, Ίντερνετ |

Fine-Tuning was performed over 312 steps, once again through Hugging Face's Trainer API and the DataCollatorForTokenClassification class provided specifically for tasks such as this. Evaluation was performed every 70 steps on the development set. The following arguments were used.

**Table 4: Fine-Tuning Arguments for the PoS Tagging Task**

| **Argument** | **Value** |
|---|---|
| per_device_train_batch_size | 16 |
| per_device_train_eval_size | 16 |
| learning_rate | 1e-4 |
| weight_decay | 1e-4 |
| eval_steps | 70 |

Fine-Tuning was completed in 61 seconds. After the training process, validation loss was at 0.09. The trained weights were then used to predict the classes of the unseen test set. Test loss was at 0.08, and test accuracy was at 97.7%, incredibly close to GREEK-BERT's 98.1%. Thus, DistilBERT's promise of retaining at least 97% of its larger counterpart's accuracy is proven very true for this task. The board below indicates the per-class performance score in terms of precision, recall and F1.

**Table 5: Per-Class Evaluation Metrics on the PoS Tagging task**

| Label | Precision | Recall | F1 Score |
|---|---|---|---|
| **ADJ** | 0.942 | 0.963 | 0.952 |
| **ADP** | 0.998 | 0.995 | 0.997 |
| **ADV** | 0.954 | 0.969 | 0.961 |
| **AUX** | 0.998 | 1.000 | 0.999 |
| **CCONJ** | 0.984 | 0.994 | 0.989 |
| **DET** | 0.996 | 0.997 | 0.997 |
| **NOUN** | 0.975 | 0.978 | 0.977 |
| **NUM** | 0.927 | 0.897 | 0.912 |
| **PART** | 1.000 | 1.000 | 1.000 |
| **PRON** | 0.994 | 0.973 | 0.983 |
| **PROPN** | 0.850 | 0.837 | 0.843 |
| **PUNCT** | 1.000 | 1.000 | 1.000 |
| **SCONJ** | 1.000 | 0.989 | 0.994 |
| **VERB** | 0.996 | 0.987 | 0.991 |
| **X** | 0.752 | 0.693 | 0.721 |
| **macro avg** | 0.958 | 0.951 | 0.954 |
| **weighted avg** | 0.977 | 0.977 | 0.977 |

The following board compares the 2 models with regard to their F1 scores per class.

**Table 6: Head To Head Comparison on the PoS Tagging Task**

| Label | GREEK-BERT | DistilBERT-EL-KLD |
|---|---|---|
| **ADJ** | 95.6 | 95.2 |
| **ADP** | 99.7 | 99.7 |
| **ADV** | 97.2 | 96.1 |
| **AUX** | 99.9 | 99.9 |
| **CCONJ** | 99.6 | 98.9 |
| **DET** | 99.8 | 99.7 |
| **NOUN** | 97.9 | 97.7 |
| **NUM** | 92.7 | 91.2 |
| **PART** | 100.0 | 100.0 |
| **PRON** | 98.8 | 98.3 |
| **PROPN** | 86.0 | 84.3 |
| **PUNCT** | 100.0 | 100.0 |
| **SCONJ** | 99.4 | 99.4 |
| **VERB** | 99.3 | 99.1 |
| **X** | 77.3 | 72.1 |

Aside from the **X** class, which does not refer to regular Greek vocabulary words, F1 scores appear to be very close to each other, constantly maintaining a gap of less than 2%. The models also seem to have identical performance on classes such as SCONJ, PART and ADP. Also, DistilBERT's (61 seconds) runtime is significantly faster when compared to that of GREEK-BERT (3 minutes).

### 3.2.3 Named Entity Recognition

The task of tagging named entities proved to be quite harder for both the original GREEK-BERT model as well as its student. For the purposes of this task, we had to utilize a combination of 2 datasets with completely different formats, thus requiring significantly different preprocessing pipelines. The first dataset, created by I. Darras [8], comes in JSONL format. From all the different keys located in each individual JSON, we only elected to keep *spans* and *text*. The spans element contained the

label of each NER in question, the starting and end position of the entity in the text (measured in characters), as well as an “answer” element, taking the values of either *accept* or *ignore*. The “ignore” tag seemed to be reserved for cases where the named entity was wrongly tagged, for example there were cases where a comma character was tagged as a person, location or organization. After selecting the following elements, the initial dataset (after retrieving the named entity text through the given indices) looked like this:

| | text | tokens | answer | label | start | end | entity |
|---|---|---|---|---|---|---|---|
| 0 | Ο Νίκος Γρυλλάκης ήταν δημοσιογράφος με ήθος κ... | [{'text': 'Ο', 'id': 0, 'start': 0, 'end': 1},... | accept | PERSON | 2.0 | 17.0 | Νίκος Γρυλλάκης |
| 1 | Και έτσι μεταφέρθηκε στον Σέρχιο Αγουέρο όταν ... | [{'text': 'Και', 'id': 0, 'start': 0, 'end': 3... | accept | PERSON | 26.0 | 40.0 | Σέρχιο Αγουέρο |
| 2 | Ο επιθετικός της Μάντσεστερ Σίτι αιφνιδιασμένο... | [{'text': 'Ο', 'id': 0, 'start': 0, 'end': 1},... | accept | ORG | 17.0 | 32.0 | Μάντσεστερ Σίτι |
| 3 | Ήδη από το 1947 ο Ντιόρ είχε παρουσιάσει το πρ... | [{'text': 'Ήδη', 'id': 0, 'start': 0, 'end': 3... | accept | PERSON | 18.0 | 23.0 | Ντιόρ |
| 4 | Ήδη από το 1947 ο Ντιόρ είχε παρουσιάσει το πρ... | [{'text': 'Ήδη', 'id': 0, 'start': 0, 'end': 3... | accept | GPE | 88.0 | 95.0 | Λονδίνο |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 374 | Η πίεση από τον Αργεντινό άσο μάλλον επηρέασε ... | [{'text': 'Η', 'id': 0, 'start': 0, 'end': 1},... | accept | PERSON | 164.0 | 169.0 | Βαράν |
| 375 | Η πίεση από τον Αργεντινό άσο μάλλον επηρέασε ... | [{'text': 'Η', 'id': 0, 'start': 0, 'end': 1},... | accept | PERSON | 195.0 | 199.0 | Μέσι |
| 376 | Η πίεση από τον Αργεντινό άσο μάλλον επηρέασε ... | [{'text': 'Η', 'id': 0, 'start': 0, 'end': 1},... | accept | PERSON | 234.0 | 241.0 | Μαρσέλο |
| 377 | Η πίεση από τον Αργεντινό άσο μάλλον επηρέασε ... | [{'text': 'Η', 'id': 0, 'start': 0, 'end': 1},... | accept | PERSON | 250.0 | 263.0 | Τζόρντι Αλμπα |
| 378 | Στη διασταύρωση της Εκτενεπόλ στο εμπορικό κέν... | [{'text': 'Στη', 'id': 0, 'start': 0, 'end': 3... | accept | GPE | 20.0 | 29.0 | Εκτενεπόλ |

**Figure 19: Preprocessed sample of the Greek SpaCy NER corpus**

The second dataset, compiled by A. Romanou[2], came in a CONLL-U format not dissimilar to the UD dataset examined in the previous task. We elected to keep the tokens of each individual text, as well as the NameType element along with the corresponding tag.

```
# sent_id = gdt-20020204-ep-sessions_044-9
# text = Εκφράζουμε την ικανοποίησή μας με τους στόχους της έκθεσης του κ. Oostlander.
1       Εκφράζουμε      εκφράζω VERB    VERB    Aspect=Imp|Mood=Ind|Number=Plur|Person=1|Tense=Pres|VerbForm=Fin|Voice=Act      0       root    _       _
2       την     ο       DET     DET     Case=Acc|Definite=Def|Gender=Fem|Number=Sing|PronType=Art        3       det     _       _
3       ικανοποίησή     ικανοποίηση     NOUN    NOUN    Case=Acc|Gender=Fem|Number=Sing 1       obj     _       _
4       μας     μου     PRON    PRON    Case=Gen|Gender=Masc|Number=Plur|Person=1|Poss=Yes|PronType=Prs 3       nmod    _       _
5       με      με      ADP     ADP     _       7       case    _       _
6       τους    ο       DET     DET     Case=Acc|Definite=Def|Gender=Masc|Number=Plur|PronType=Art       7       det     _       _
7       στόχους στόχος  NOUN    NOUN    Case=Acc|Gender=Masc|Number=Plur        3       nmod    _       _
8       της     ο       DET     DET     Case=Gen|Definite=Def|Gender=Fem|Number=Sing|PronType=Art        9       det     _       _
9       έκθεσης έκθεση  NOUN    NOUN    Case=Gen|Gender=Fem|Number=Sing 7       nmod    _       _
10      του     ο       DET     DET     Case=Gen|Definite=Def|Gender=Masc|Number=Sing|PronType=Art       11      det     _       _
11      κ.      κ.      NOUN    NOUN    Abbr=Yes        9       nmod    _       _
12      Oostlander      Oostlander      X       X       Foreign=Yes|NameType=PERSON     11      flat    _       SpaceAfter=No
13      .       .       PUNCT   PUNCT   _       1       punct   _       _
```

**Figure 20: Sample from the NER Dataset by A. Romanou**

The dataset provided by Romanou was split into train, dev and test files. Darras’ dataset was split into 3 files, also used as training, dev and test sets. After removing duplicates, and keeping only entities with the tags LOC, PERSON, GPE (them being the only common ones across the 2 datasets, LOC labels were renamed to GPE because they technically represent the same objects), the final dataset used for training and evaluation contained a training set of 1,622 sentences, the development set of 404 sentences, and finally the test set of 456 sentences. Essentially the label set consisted of 7 classes, each of them having the suffix **B-** when being the first token in a named entity, or **I-** in all other cases (IOB scheme).

[2] Special Thanks to Despina-Athanasia Pantazi for the provision of the dataset, since it is no longer available online.

**Table 7: List of Labels Used for the NER Task**

| | |
|---|---|
| **B-ORG, I-ORG** | Organizations, such as institutions, sports teams, companies etc: Παναθηναϊκός, Microsoft, Ευρωπαϊκή Ένωση |
| **B-GPE, I-GPE** | Geopolitical Entities, i.e. countries, continents, cities: Αθήνα, Ελλάδα, Ευρώπη |
| **B-PERSON, I-PERSON** | Names belonging to people: Παπαδόπουλος, Van Gogh |
| **O** | Not a named entity. Ignored during the evaluation process. |

Training was performed over 208 steps (2 epochs), with 2 intermittent evaluation steps, one at step 104, and one at the end. Fine-tuning lasted about 140 seconds.The following table describes the training arguments used on the Trainer API.

**Table 8: Fine-Tuning Arguments for the NER Task**

| **Argument** | **Value** |
|---|---|
| per_device_train_batch_size | 16 |
| per_device_eval_batch_size | 16 |
| num_training_epochs | 2 |
| learning_rate | 1e-4 |
| weight_decay | 1e-4 |
| evaluation_strategy | “steps” |
| eval_steps | 104 |

At the end of the training process, validation loss was about 0.35. On the unseen test set, overall F1 score was at 89%, surpassing the original GREEK-BERT’s 85.7. This can be attributed to different methods of preprocessing used by the original GREEK-BERT research team, which could have led to the inclusion of some data that could be considered noise (recall that certain commas were tagged as named entities).

**Table 9: Per-class Evaluation Metrics for the NER Task**

| **Label** | Precision | Recall | F1 |
|---|---|---|---|
| **test_GPE** | 0.917 | 0.934 | 0.926 |
| **test_ORG** | 0.825 | 0.801 | 0.813 |
| **test_PERSON** | 0.878 | 0.894 | 0.886 |
| **macro_avg** | 0.887 | 0.893 | 0.890 |

In order to increase fairness, a class-based metrics comparison will be made between the current model, as well as the original GREEK–BERT architecture fine-tuned on the current iteration of the dataset.

**Table 10: Head To Head Comparison on the NER Task**

| **Label** | **DistilBERT-EL-KLD** | **GREEK-BERT (fine-tuned)** | **GREEK-BERT (original)** |
|---|---|---|---|
| **GPE** | 92.6 | 93.7 | 88.8 |
| **PERSON** | 88.6 | 93.4 | 88.4 |
| **ORG** | 81.3 | 86.8 | 69.6 |

It seems that our model is lacking only in predicting ORG values, which could be attributed to a lot of them encapsulating instances of the other 2 labels (for example Μάντσεστερ should be tagged as a GPE, but Μάντσεστερ Γιουνάιτεντ as an ORG) [16]. Either way overall accuracy for the new GREEK-BERT iteration is 96.4%, and a 2% overall gap between the 2 models is completely logical. Our model still displays the superior runtime, running over the course of 2 minutes and 20 seconds, while the original Greek-BERT architecture takes about 4 minutes and 20 seconds to complete training.

### 3.2.4 Natural Language Inference

For the final task of Natural Language Inference (NLI), we utilized the Greek portion of the MultiNLI dataset, a multilingual version of the English XNLI corpus [2]. Data preparation and preprocessing was less complicated than that of the previous 2 tasks, since the dataset comes in .tsv format. The task is to predict whether a sequence of 2 sentences (a premise and a hypothesis) expresses some kind of entailment or contradiction (if any, if not the relationship is tagged as neutral). This is a sequence classification task as opposed to the previous ones, them being examples of token classification, and having to do with the logic behind a sequence of sentences, it could be seen as a task not too dissimilar to the Next Sentence

Prediction (NSP) task. The significant downside of the dataset is that the multilingual corpus was generated through NMT methods applied on the original English corpus. That means that the quality of some of the sentences could be dubious and very divergent from the intended translations when generated by actual human native speakers.

| | premise | hypo | label |
|---|---|---|---|
| 0 | η εννοιολογικα κρεμα κρεμα εχει δυο βασικες δι... | το προϊον και η γεωγραφια ειναι αυτα που κανο... | neutral |
| 1 | ξερεις κατα τη διαρκεια της σεζον και υποθετω ... | χανεις τα πραγματα στο επομενο επιπεδο , αν οι... | entailment |
| 2 | ενας απο τους αριθμους μας θα μεταφερει τις οδ... | ενα μελος της ομαδας μου θα εκτελεσει τις διατ... | entailment |
| 3 | πως το ξερεις ; ολες αυτες ειναι οι πληροφοριε... | αυτες οι πληροφοριες ανηκουν σ ' αυτους . | entailment |
| 4 | ναι , θα σου πω τι θα γινει αν πας να παρεις μ... | τα παπουτσια του τενις εχουν μια σειρα απο τιμ... | neutral |
| ... | ... | ... | ... |
| 392697 | προφανως , η καλιφορνια μπορει - και πρεπει - ... | η καλιφορνια δεν μπορει να κανει κατι καλυτερο . | contradictory |
| 392698 | καποτε θεωρηθηκε ο πιο ομορφος δρομος στην ευρ... | πολλα απο τα αρχικα κτιρια ειχαν αντικατασταθε... | neutral |
| 392699 | οι πλοια ειναι μια ωραια διατηρημενα παραδοση ... | η παραδοση του πλοια ξεκινησε ενω ο βρετανος ρ... | entailment |
| 392700 | οι νεκρολογιες υπενθυμισαν με αγαπη τις συζητη... | οι νεκρολογιες ηταν ομορφες και γραμμενες σε ε... | neutral |
| 392701 | σε αυτο το αλλο ξερεις οτι πρεπει να το κανω η... | ο αντρας μου εχει τοση πολλη δουλεια τελευταια... | neutral |

392702 rows × 3 columns

**Figure 21: Example of the Greek Portion of the MultiNLI corpus**

The dataset was split into training, development and validation sets. The development and validation sets had multiple label annotations for every given sequence (5 in number). In every case, the label that had the majority vote was selected, and if 2 labels had the same amount of votes, the one that appeared first was selected. In the development and test sets, there was also a column *match* indicating whether the produced sentences in the sequence actually matched or not (some of them, when put in a sequence did not make sense at all and were filtered out).There The training set was composed of 392702 training samples, the development set consisted of 2312 samples, and the test set consisted of 4666 samples. Training lasted for exactly 2 hours, performing 24544 training steps over 2 epochs and hyperparameters during training were the following:

**Table 11: Fine-Tuning arguments for the NLI Task**

| **Argument** | **Value** |
|---|---|
| per_device_train_batch_size | 32 |
| per_device_eval_batch_size | 32 |
| num_train_epochs | 2 |
| learning_rate | 1e-4 |
| weight_decay | 1e-4 |
| evaluation_strategy | “steps” |
| evaluation_steps | 1000 |
| hidden_dropout_prob | 0.2 |
| attention_probs_dropout_prob | 0.2 |

The dropout probability of the initial model configuration was raised to 0.2 from the original 0.1, since it helped in the accuracy metric increasing by 1-2%. Validation and Test loss were both at around 0.63. Evaluation was performed by using simple classification metrics such as precision, recall and F1 score (macro average) functions - both overall and per class- provided by the scikit-learn library. An overall accuracy of 73.5 shows, for the first time, a larger gap between our model and the original GREEK-BERT, which attained an accuracy of 78.6. That 5% gap could either be attributed to the poor quality of the training data, being an AI-generated dataset, but it might also signify the need of knowledge distillation training in the Next Sentence Prediction task, through which out model might be able to draw patterns regarding longer-range (intersentential) dependencies.

**Table 12: Per-Class evaluation metrics for the NLI Task**

| **Label** | **Precision** | **Recall** | **F1** |
|---|---|---|---|
| Neutral | 68.9 | 72.5 | 70.1 |
| Entailment | 80.0 | 69.8 | 72.6 |
| Contradiction | 74.7 | 80.5 | 76.8 |
| macro_avg | 74.5 | 74.2 | 74.2 |

A closer look at the performance (with regards to the F1-score) of both models might shed more light on the matter.

**Table 13: Head To Head Comparison on the NLI Task**

| Label | DistilBERT-EL-KLD | Greek-BERT |
|---|---|---|
| **Neutral** | 70.1 | 75.9 |
| **Entailment** | 72.6 | 78.8 |
| **Contradiction** | 76.8 | 81.2 |

Evidently, just like the teacher before it, the student model generalizes better on sequences that imply contradiction and falls short when the relationship is completely neutral. Our model is constantly about 5% behind, but even so the performance, considering the difficulty of the task as well as the data at hand is adequate, especially given that during knowledge distillation a certain portion of the teacher model's performance has to be downsized. The only clear advantage is that of training speed, since the original GREEK-BERT has to be trained for about 3 hours in order to perform the same task.

### 3.3 Summary

The full development of DistilBERT-EL-KLD was thoroughly examined: from the selection of the pretraining corpora, to preprocessing, to MLM-based pretraining with Knowledge Distillation and finally fine-tuning on NLU tasks, such as PoS-Tagging, NER and NLI. The model appears to be significantly faster and computationally inexpensive than its decompressed counterpart, while also appearing to preserve the original model's accuracy on all tasks, except maybe on the NLI task, even though it does not fail in achieving a quite satisfactory performance in general.

# 4. CONCLUSION

The experiments conducted above were able to provide a more thorough understanding on the inner workings of language models. Even when aiming at reducing parameter size and computational needs, one has to take into account that the undertaking still has very considerable requirements, concerning time and resources alike. But the argument still remains: not every single natural language understanding or generation task requires a large language model. Training and evaluation of the DistilBERT-EL-KLD model have demonstrated that smaller models can be pretrained on a relatively small amount of data, also allowing for faster deployment and evaluation, while retaining human-level performance for a plethora of tasks. Any kind of initial inefficiency can be mitigated through further pretraining or fine-tuning with a specific purpose, but state-of-the art performance has also shown that even on fine-tuning tasks, the quality of the input data plays a paramount role on the model's performance. With time, larger datasets and even higher quality data will be available in many written natural languages, thus leading to even more improvement in model inference and evaluation. The goal of the democratization of AI seems now more attainable than ever, especially for simple tasks like text classification. With smaller architectures enabled by Knowledge Distillation, as well as repositories such as Hugging Face, the future looks very bright for the field, and one might even entertain the thought that given a home computer and an internet connection, anyone can develop their very own language model.

## LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| API | Application Programming Interface |
| BERT | Bidirectional Encoding Representations from Transformers |
| CoNLL | Computational Natural Language Learning |
| DNN | Deep Neural Network |
| GPT | Generative Pretrained Transformer |
| GRU | Gated Recurrent Unit |
| HPC | High Performance Computer |
| I-O-B | Inside-Outside-Beginning |
| KLD | Kullback-Leibler Divergence |
| LLM | Large Language Model |
| LSTM | Long Short Term Memory |
| ML | Machine Learning |
| MLM | Masked Language Modeling |
| NER | Named Entity Recognition |
| NLG | Natural Language Generation |
| NLI | Natural Language Inference |
| NLP | Natural Language Processing |
| NLU | Natural Language Understanding |
| NMT | Neural Machine Translation |
| NSP | Next Sentence Prediction |
| PoS | Part of Speech |
| ReLU | Rectified Linear Unit |
| RNN | Recurrent Neural Network |

# LIST OF REFERENCES


[1] F. Chollet, *Deep Learning with Python*, Manning, 2021.

[2] A. Conneau et al., *XNLI: Evaluating Cross-lingual Sentence Representations*, in: E. Riloff, D. Chiang, Julia Hockenmaier, J. Tsujii (eds.), Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2475-2485.

[3] M. P. Deisenroth, A. A. Faisal, C. S. Ong, *Mathematics for machine learning*, Cambridge University Press.

[4] J. Devlin, M. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, in: J. Burstein, C. Doran, and T. Solorio (eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171-4186.

[5] D. Foster, *Generative Deep learning: Teaching Machines to Paint, Write, Compose, and Play*, O'Reilly, 2023.

[6] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly, 2019.

[7] I. Goodfellow, Y. Bengio, A. Courville, *Deep learning*, MIT Press, 2016.

[8] Google Summer of Code 2018 Project - spaCy now speaks Greek, GitHub, https://github.com/eellak/gsoc2018-spacy [Online; Accessed 15/03/2024]

[9] J. Grus, Data *Science From Scratch: First Principles with Python*, O'Reilly, 2019.

[10] G. E. Hinton, O. Vinyals, J. Dean, Distilling the Knowledge in a Neural Network, ArXiv (Cornell University), 2016.

[11] Hugging Face. (n.d.). Hugging Face – On a mission to solve NLP, one commit at a time, [Online; Accessed 15/03/2024]

[12] C. Huyen, *Designing machine learning systems: An Iterative Process for Production-Ready Applications*, O'Reilly, 2022.

[13] G. James, D. Witten, T. Hastie, R. Tibshirani, J. Taylor, *An introduction to statistical learning: with Applications in Python*, Springer, 2013.

[14] D. Jurafsky, J. H. Martin, *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Pearson, 2014.

[15] P. Koehn, Europarl: *A Parallel Corpus for Statistical Machine Translation, in: Conference Proceedings: the tenth Machine Translation Summit*, Asia-Pacific Association for Machine Translation, Phuket, Thailand, 2005, pp. 79–86.

[16] J. Koutsikakis, I. Chalkidis, P. Malakasiotis, I. Androutsopoulos. GREEK-BERT: the Greeks visiting Sesame street, in: Constantine D. Spyropoulos, I. Varlamis, I. Androutsopoulos, and P. Malakasiotis,(eds.), 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, 2020.

[17] Y. Liu, M. Ott, N. Goyal, J. Du, M. S. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, ArXiv (Cornell University), 2019 pp. 110–117.

[18] L. Martin, B. Muller, P. J. O. Suárez, Y. Dupont, L. Romary, É. V. de la Clergerie, D. Seddah, & B. Sagot, *CamemBERT: a Tasty French Language Model*, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020, pp. 7203–7219.

[19] P.R Norvig, S. Russell, *Artificial intelligence: A Modern Approach, Global Edition*, Pearson, 2021.

[20] P. Prokopidis, E. Desypri, M. Koutsombogera, H. Papageorgiou, and S. Piperidis, T*heoretical and Practical Issues in the Construction of a Greek Dependency Treebank*, in: Proceedings of The Fourth Workshop on, Treebanks and Linguistic Theories (TLT 2005), M. Civit, S. K., and M. A. Marti (eds.). Universitat de Barcelona, Barcelona, Spain, 2005, pp. 149–160.

[21] D. Rothman, A. Gulli, *Transformers for Natural Language Processing,* Packt, 2022.

[22] A. Rush, *The Annotated Transformer*, in: Proceedings of Workshop for NLP Open Source Software, Melbourne, Australia, Association for Computational Linguistics, 2018, pp. 52-60.

[23] V. Sanh, L. Debut, J. Chaumond, T. Wolf, *DistilBERT, A Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter*, ArXiv (Cornell University), 2019.

[24] OSCAR: A Clean Version of the Common Crawl, https://oscar-project.org/ [Online; Accessed 18/03/2024]

[25] R. S Sutton, & A. Barto, *Reinforcement learning: an introduction*, MIT Press, 2018.

[26] L. Tunstall, L. von Werra, T. Wolf, *Natural Language Processing with Transformers, Revised Edition*, O'Reilly, 2022.

[27] A. M. Turing, Computing Machinery and Intelligence, in: Mind vol. 49, 1950, pp. 433–460.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is All You Need*, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4, Long Beach, CA, USA, 2017, pp. 5998–6008.

[29] Wikipedia, Hellenic languages — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Hellenic%20languages&oldid=1033038922, 2024. [Online; Accessed 15/03/2024]

[30] G. Yenduri. M. Ramalingam, C. S. G, Y. Supriya, G. Srivastava, P. K. R. Maddikunta, D. R. G, R. H. Jhaveri, B. Prabadevi, W. Wang, A. V Vasilakos, T. R. Gadekallu, G*enerative Pre-trained Transformer: a Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future directions*, ArXiv (Cornell University), 2023.