ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Bachelor Thesis
in
Computer Science

A Transformer-based Natural Language Processing Toolkit for Greek – Named Entity Recognition and Multi-task Learning

Nikolaos Smyrnioudis

Supervisors: Ion Androutsopoulos
John Koutsikakis

July 2021

Nikolaos Smyrnioudis

A Transformer-based Natural Language Processing Toolkit for Greek – Named Entity Recognition and Multi-task Learning

July 2021

Supervisors: Ion Androutsopoulos, John Koutsikakis

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Natural Language Processing Group

Athens, Greece

Abstract

Named Entity Recognition (NER), Part-of-Speech (POS) tagging and Dependency Parsing are tasks that are central to Natural Language Processing (NLP) because they provide important information that can be used for other higher level NLP tasks. We develop a series of Deep Learning models that can perform these tasks in Greek. We experiment with GreekBERT, a pre-trained Transformer model for Greek, and XLM-R, a multi-lingual pre-trained Transformer model, providing additional training to (fine-tuning) both models in Greek for the tasks we consider. We find that using GreekBERT leads to better results in Dependency Parsing. In NER and POS tagging, GreekBERT and XLM-R lead to similar results. This work was combined with the work of the BSc thesis of Chrysa Dikonimaki [Dik21]. The two theses jointly developed a publicly available Transformer-based toolkit for Greek NLP. In this thesis, the emphasis was mostly in NER and multi-task learning experiments that jointly considered all three tasks.

Περίληψη

Η αναγνώριση ονομάτων οντοτήτων (Named Entity Recognition), η κατηγοριοποίηση λέξεων σε μέρη του λόγου (Part-of-Speech Tagging, POS tagging) και η συντακτική ανάλυση εξαρτήσεων (Dependency Parsing) είναι προβλήματα πολύ κεντρικά στην επεξεργασία φυσικής γλώσσας (ΕΦΓ, Natural Language Processing, NLP) γιατί η επίλυση τους μας προσφέρει πολύτιμη πληροφορία που μπορεί να χρησιμοποιηθεί σε άλλα προβλήματα ΕΦΓ. Σε αυτή την εργασία αναπτύσσουμε μια σειρά μοντέλων βασισμένων στην Βαθιά Μάθηση (Deep Learning) για την επίλυση των συγκεκριμένων προβλημάτων στην ελληνική γλώσσα. Πειραματιζόμαστε με το GreekBERT, ένα προ-εκπαιδευμένο μοντέλο Transformer για την ελληνική γλώσσα, καθώς και με το XLM-R, ένα πολύγλωσσο προ-εκπαιδευμένο μοντέλο Transformers, παρέχοντας συμπληρωματική εκπαίδευση (fine-tuning) και στα δύο μοντέλα για τα προβλήματα που εξετάζουμε. Η χρήση του GreekBERT οδηγεί σε καλύτερα αποτελέσματα στη συντακτική ανάλυση εξαρτήσεων. Στην αναγνώριση ονομάτων οντοτήτων (NER) και στην κατηγοριοποίηση σε μέρη του λόγου (POS tagging), το GreekBERT και το XLM-R οδηγούν σε παρόμοια αποτελέσματα. Αυτή η εργασία συνδυάστηκε με την πτυχιακή εργασία της Χρύσας Δικονιμάκης [Dik21]. Οι δύο εργασίες ανέπτυξαν από κοινού μια δημόσια διαθέσιμη εργαλειοθήκη ΕΦΓ για την ελληνική γλώσσα, βασισμένη σε Transformers. Στην παρούσα εργασία, η έμφαση δίδεται στην αναγνώριση ονομάτων οντοτήτων, καθώς και σε πειράματα από κοινού μάθησης (multi-task learning) και για τα τρία προβλήματα μαζί.

Acknowledgements

I would like to thank everyone that has supported me with materials, advice and resources during the work of this thesis. I would like to especially thank Professor Ion Androutsopoulos for providing me with important ideas and learning resources for my thesis. Many thanks also go to John Koutsikakis for assisting with the technicalities of implementing Deep Learning models with PyTorch and continuous advice and supervision throughout my work. Many thanks also to Katerina Korre, for proof-reading and writing advice, to Manolis Kyriakakis for providing code and guidance with the implementation of the Dependency Parsing model and to Mary Georgiou for assisting with the development of the toolkit.

Contents

1 Introduction

1.1 What is the problem we solve?

In this thesis we worked on Named Entity Recognition (NER), Part of Speech (POS) Tagging and Dependency Parsing (DP) for the Greek Language.

NER is the task of locating named entities in text. These can be names of persons, organizations and many more. This task is more semantic in nature, because we need to be able to understand where the named entities are in a text as well as classify them into semantic categories.

POS tagging is the task of identifying the part of speech of each word. This can require, for example, determining which words in a sentence are verbs, nouns or articles. POS tagging can have more fine-grained tags. For inflectionally rich languages like Greek, nouns can also be characterised by their case, like nominative or accusative, and verbs by their person, for example 3rd singular or 3rd plural etc. and many other morphological categories.

DP is the task of defining the syntactic structure of a sentence. Unlike POS tagging which studies each word alone (but in its context), in DP we must define the relationships between words. For example, in the case where the verb is "reimburses", we need to find *who* reimburses (the subject of the verb) and *who* is reimbursed (the object of the verb).

1.2 Why is it important?

Information from NER, POS tagging, and DP can be useful in several other tasks, such as question answering, information extraction, and information retrieval.

1.3 What has been done until now

NER, POS and DP are already supported by toolkits like spaCy[1], NLTK[2] and Stanza[3] in many different languages. However, high performance in these tasks is very important and we can get improvements for a specific language by optimizing the models for that language.

1.4 What we did

1.4.1 Transformer models

For our models we used the GreekBERT transformer [Kou+20]. GreekBERT is an instance of the BERT [Dev+18] transformer architecture which is pretrained on exclusively Greek corpora. A very strong advantage of GreekBERT, as the authors of [Kou+20] point out, other than the extensive Greek datasets on which it was pretrained, is the lower rate of fragmentation in Greek words compared to other multilingual transformer models.

BERT models such as ours use only the encoder transformer modules introduced in [Vas+17] and are pretrained on text corpora in an self-supervised fashion. One of the self-supervised training tasks is Masked Language Modelling (MLM), in which the model, given a sentence with some words replaced with the MASK token, tries to predict the missing words. The other pre-training task is Next Sentence Prediction (NSP), in which the model given two sentences tries to predict if one sentence would follow the other in a document.

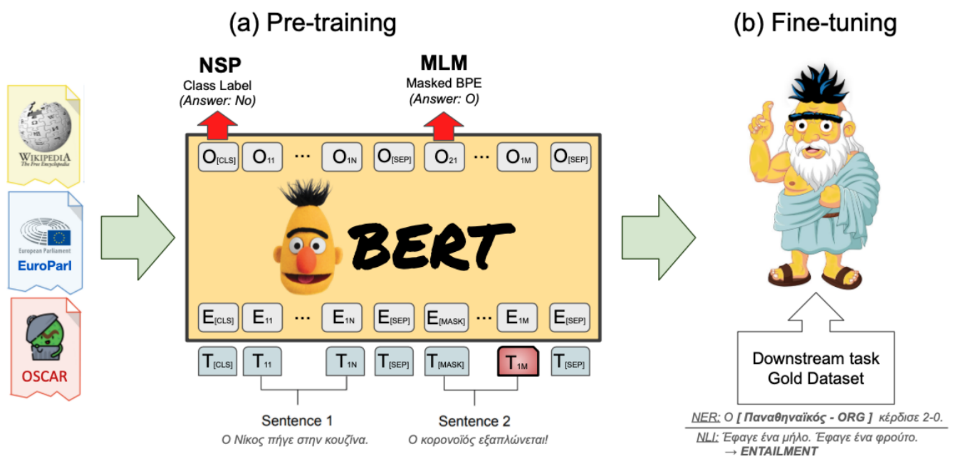
Apart from GreekBERT, for some of the experiments we used the XLM-R Transformer [Con+19]. XLM-R is also an instance of the BERT architecture but it is pretrained with text sources from multiple languages.

Using GreekBERT or XLM-R, we developed task specific prediction heads for NER, POS tagging and DP and fine-tuned each one to perform well on task specific benchmarks. The benchmarks we used is the NER dataset developed in [BMK20] and the Greek treebank from the Universal Dependencies dataset [Niv+16].

We compared our models' performance on the benchmarks mentioned with the performance of the Stanza toolkit [Qi+20] on POS tagging and DP (Stanza does not offer NER for

[1]https://spacy.io/
[2]https://www.nltk.org/
[3]https://stanfordnlp.github.io/stanza/

Fig. 1.1: An illustration of GreekBERT pretraining. Figure taken from [Kou+20].


(a) Pre-training
(b) Fine-tuning
NSP
Class Label
(Answer: No)
MLM
Masked BPE
(Answer: O)
WIKIPEDIA
EuroParl
OSCAR
BERT
$O_{[CLS]}$ O_{11} ... O_{1N} $O_{[SEP]}$ O_{21} ... O_{1M} $O_{[SEP]}$
$E_{[CLS]}$ E_{11} ... E_{1N} $E_{[SEP]}$ $E_{[MASK]}$ ... E_{1M} $E_{[SEP]}$
$T_{[CLS]}$ T_{11} T_{1N} $T_{[SEP]}$ $T_{[MASK]}$ T_{1M} $T_{[SEP]}$
Sentence 1
Ο Νίκος πήγε στην κουζίνα.
Sentence 2
Ο κορονοϊός εξαπλώνεται!
Downstream task Gold Dataset
NER: Ο [Παναθηναϊκός - ORG] κέρδισε 2-0.
NLI: Έφαγε ένα μήλο. Έφαγε ένα φρούτο. → ENTAILMENT

Segmentation			
se	##g	##ment	##ation

Tab. 1.1: A word segmented into subword tokens. The initial subword token is "se" and the non-initial subword tokens are "##g", "##ment", "##ation".

Greek). In the case of NER we compared our models' performance with the results shown in [BMK20] where different transformer models were tested.

Before moving on to the main chapters of the thesis, let us clarify how we use the subwords (Table 1.1) generated by the tokenizers of the pre-trained Transformer models we employ (GreekBERT, XLM-R). For all tasks we extract labels only for the first subword token of each word and assign these labels to the whole word. However this doesn't mean that the model will not utilize all the sub-words for its predictions. Each layer of GreekBERT or XLM-R, produces embeddings for all the sub-words, and the embedding of each sub-word is a function of the previous layer's embeddings (except the first layer where the embeddings are retrieved from a look-up table) including non-initial subword embeddings.

1.5 Thesis Structure

In 2.1 we describe our model for NER as well as the dataset and the results. In sections 3 and 4 we explore MTL (MultiTask Learning) between NER and POS as well as DP and POS. In each section we describe the problems in more detail, showing examples from the datasets, detailed descriptions of the prediction heads as well as details about the fine-tuning procedure and the final results and comparisons on the benchmarks. Finally in 5 we conclude summarizing our work and results.

2 Named Entity Recognition

2.1 Introduction

Named Entity Recognition (NER) is the task of locating named entities in some input text. The named entities are spans of words, meaning continuous words in a sentence, and can be a location, organisation or named entit1ies from many other categories. Our tagger can recognise 18 different fine-grained categories or 4 coarse-grained categories. These categories with their occurences are listed in tables 2.1 and 2.2.

2.2 Why is NER important?

When we have to deal with large datasets and unstructured data , having a way to easily and quickly extract useful information is crucial. Several NLP applications need named entities. For example, we can use them to handle customer requests faster and boost user satisfaction by identifying product names, serial numbers etc. in customer e-mail messages, user utterances in chatbot dialogues etc.

Moreover, named entities can be used to obtain feedback per person or organization, by collecting user posts that mention entities of particular types and applying sentiment analysis. NER models especially for social networks have also been developed; see [RCE+11] for a survey of NER models for tweets

Several companies use named entities in their recommendation systems in order to improve customer experience[1]. Also, an interesting application of NER is resume parsing, where NER can reduce the amount of time recruiters spend to find useful information in unstructured CVs [Tos+15].

2.3 Related work

Older approaches to NER use hand-crafted features. [MAM08] use Brown clusters, dictionary features for each word, as well as a POS tagging system as part of their pipeline.

[1]https://www.allerin.com/blog/exploring-named-entity-recognition-use-cases-across-industries

Category	Num. Occurences
ORG	13527
PERSON	10879
CARDINAL	9013
GPE	8367
DATE	7855
PERCENT	1789
ORDINAL	1774
LOC	1753
NORP	1693
TIME	1277
MONEY	1233
EVENT	1209
PRODUCT	826
WORK_OF_ART	757
FAC	722
QUANTITY	685
LAW	284
LANGUAGE	75

Tab. 2.1: The 18 semantic categories of the elNER18 dataset and their occurences (unsplitted).

Category	Num. Occurences
ORG	13527
PERSON	10879
LOC	10120
MISC	3659

Tab. 2.2: The 4 semantic categories of the elNER4 dataset and their occurences (unsplitted).

Sentence	Pos tags
Jane	B-PER
Villanueva	E-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	E-ORG
discussed	O
the	O
Chicago	S-LOC
route	O
.	O

Tab. 2.3: Example of NER (IOBES tagging) from [Jur20].

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

Tab. 2.4: The 4 coarse-grained NER categories from [Jur20].

Sentence	IOBES tags
Ήταν	O
ο	O
επικεφαλής	O
της	O
παράταξης	O
«	O
Αυτοδιοικητικό	B-ORG
Κίνημα	E-ORG
»	O
,	O
δήμαρχος	O
Αμφιλοχίας	O
,	O
Απόστολος	B-PERSON
Κοιμήσης	E-PERSON

Tab. 2.5: An example sentence encoded with the IOBES encoding. "Αυτοδοιηκιτικό τμήμα" is an organization. The first word of the organization name is marked as "B-ORG", and the final word as E-ORG. Intermediate words of an organization name would have been tagged as I-ORG. Words not belonging to any named entity span as well as punctuation marks are labeled as "O".

For predictions they use CRF [LMP01]. More recently the neural architectures used for NER were LSTM [HS97] and variants. [Lam+16] compare vanilla LSTMs as well as stacked LSTMS with a CRF prediction head. More recently, Transformer models [Vas+17] have also been used in NER [Yam+20] [SSH19]. [BMK20] create a new Greek NER dataset and compare various transformer models as well as ELMo on this new dataset.

2.4 Dataset

NER datasets contain the text, the named entity spans and the categories of the spans. There are many encoding schemes for this information, some examples of which are XML and other markup languages. However the most common ones are IOB encodings [RM95] and variations. Our toolkit deals with IOBES encodings. In IOBES encodings a span of words is annotated as follows: if the span consists of only one word it is annotated as S-X where X the category of the span. If the span is more than one word, the first word of the span is annotated as B-X, the end word as E-X and all words in between as I-X. Lastly, the tag O is assigned to words that are outside of any span. An example encoding is presented in Table 2.3 as well as in Table 2.5.

The dataset we have been experimenting on [BMK20] [2] contains 18 different categories. It was initially annotated using an existing NER tagger [3] that used 6 categories and was further processed with the Prodigy tool [4], an active learning tool for manually annotating text corpora, to be enhanced with 12 more categories. The same dataset has also been released with 4 categories by grouping the initial 18 categories to more generic ones.

2.5 Model

Our model for named entity recognition works as follows: Every input sentence is tokenized and then the GreekBERT or XLM-R model generates embeddings for each token. Then every embedding is passed through a linear layer that outputs a vector containing logits for each class for the token. More formally for input tokens $x_i, i = 0, \ldots, n$, GreekBERT or XLM-R generates embeddings $e_i, i = 0, \ldots, n$, where for every i, $e_i \in \mathbf{R}^{768}$. Finally these embeddings are transformed to $y_i = Ae_i + b \in \mathbf{R}^C, i = 0, \ldots, n$ where C is the number of possible classes for each token. A and b are parameters that the model learns and $A \in \mathbf{R}^{C\times 768}, b \in \mathbf{R}^C$. The predicted classes will be the ones with the greatest score for every token.

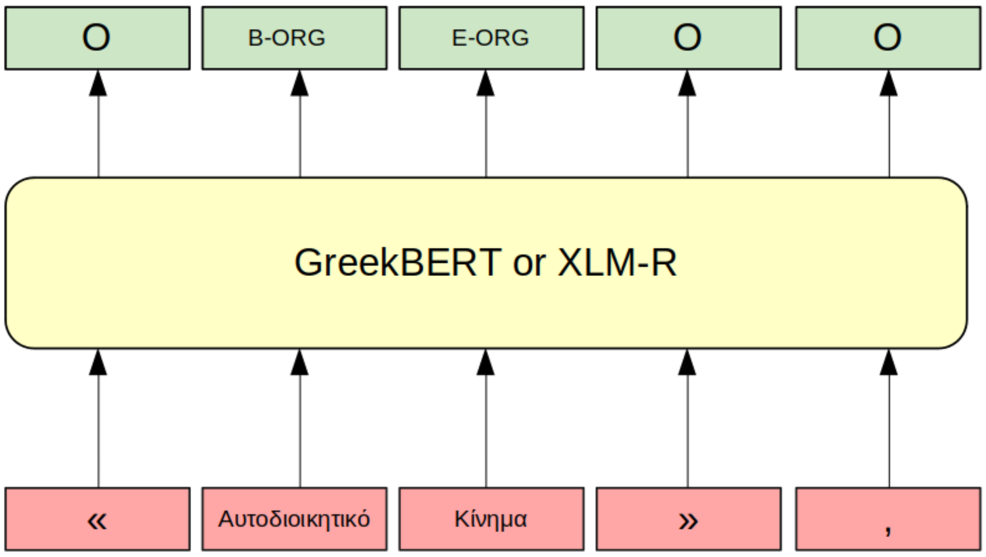
The possible classes of the token are all the possible tags in the IOBES encoding of the dataset. It is noteworthy that not every named entity category has all possible encoding tags. For example in the training, validation and test sets the entity LANGUAGE has only appeared with the tag S-LANGUAGE due to the fact that all languages are written as single words.

The predictions of the model are then gathered and with the seqeval library [5] they are chunked to extract the output spans and then the f1-scores are computed. The f1-scores are computed as follows: For every sentence the gold spans and output spans are represented as a triple (e, x, y) where e is the entity tag of the span and x , y the indices of the start and end word. If an output span is contained in the gold spans then it counts as one true positive. If an output span is not contained in the gold spans it counts as a false positive. There is also the case where a gold span is not contained in the outputs spans which counts as a false negative.

An example of this scoring can be seen in Figure 2.2. In the figure the gold spans are (PER, 1, 2), (LOC, 6, 6) and the predicted spans are (PER, 2, 3), (LOC, 6, 6). From this sentence a false positive, a false negative is counted for the Person category and a true positive is

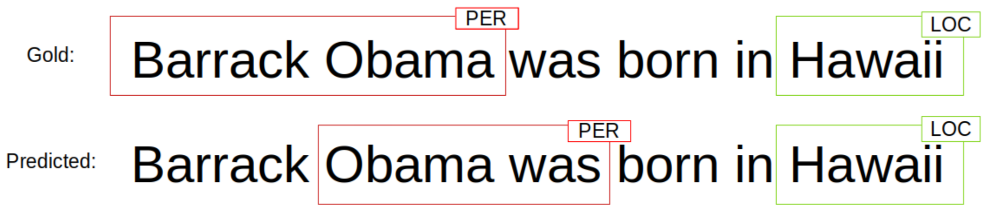
[2]The dataset can be found here https://github.com/nmpartzio/elNER
[3]https://github.com/eellak/gsoc2018-spacy
[4]https://prodi.gy/
[5]https://pypi.org/project/seqeval/

Fig. 2.1: An illustration of the NER model. Input tokens are fed to GreekBERT or XLM-R and the outputs are class labels for every token.



counted for the Location category. If the sentence was the only one in the test dataset the f1-scores would be 1 for LOC and 0 for PER.

Fig. 2.2: An example application of the entity scoring.


Gold:
PER
LOC
Barrack Obama was born in Hawaii
Predicted:
PER
LOC
Barrack Obama was born in Hawaii

2.6 Experimental Results

In our experiments we compared the model we described above with the same model but with an additional CRF [LMP01] layer (prediction head) on top, instead of the linear layer discussed above. We also compared with the best results of [BMK20] who compared many different models such as BiLSTMs, CNNs with and without a CRF prediction head and with different embeddings such as ELMo, BERT and fastText. Finally we also experimented with XLM-R for producing the embeddings.

Learning rate	Dropout	Grad accumulation steps
[5e-5, 3e-5, 2e-5]	[0, 0.1, 0.2]	[4, 8]

Tab. 2.6: Hyperparameters that were tested while tuning.

2.6.1 Training details

We tuned the models for the different hyperparameters in table 4.2 and optimized with grid search the macro-f1 score on the development subset. Early stopping was also used on macro-f1 in order to avoid overfitting. For training we used the AdamW optimizer from PyTorch [LH17] and the cross-entropy loss as the training objective.

2.6.2 Results

	macro-f1	micro-f1
GreekBERT	0.86	0.91
GreekBERT-CRF	0.84	0.90
[BMK20]-BEST	-	0.91
XLM-R	**0.87**	**0.92**

We notice that XLM-R obtains the best results. However GreekBERT performs only marginally worse than XLM-R. We also notice that using a CRF harms the performance as noted by the drop of f1-scores in the GreekBERT-CRF system.

Finally in Table 2.6.2 we also compare our GreekBERT based NER tagger with the NER systems of spaCy[6]. The lg pipeline of spaCy has the largest and most complex model while the sm pipeline has only small rule-based models.

Category	GreekBERT	spaCy-lg	spaCy-md	spaCy-sm
EVENT	0.64	0.31	0.29	0.16
GPE	0.93	0.77	0.78	0.65
PERSON	0.96	0.82	0.81	0.68
LOC	0.80	0.01	0.00	0.02
ORG	0.88	0.65	0.64	0.58
PRODUCT	0.75	0.27	0.21	0.21

We notice that GreekBERT strongly outperforms spaCy on these categories. An interesting disparity in performance is in the LOC category. Specifically, out of 175 sentences where

[6]We experimented with all the pipelines here https://spacy.io/models/el

LOC appeared in the test dataset, spaCy-lg only decided a LOC entity exists twice. Furthermore, a lot of entities whose true labels were LOC were annotated as GPE by spaCy. The situation is similar in the other spaCy pipelines. We hypothesize that the low performance of spaCy on this dataset can be attributed to the different annotation systems as well as the different domain of the annotated text sources. It would be interesting to also compare these performance on a test subset of the dataset that spaCy was trained on, but we leave that for future work.

3 Multitask Named Entity Recognition - Part of Speech Tagging

3.1 Introduction

Apart from NER, we have also developed a model that can predict NER tags as well as fine-grained Part-of-Speech (POS) tags and morphosyntactic categories (fine-grained POS tagging) which is trained with multitask learning. NER and POS tasks are often combined and predictions are offered in natural language processing pipelines such as SpaCy[1] and NLTK[2] . Both tasks are classified as sequence-labeling tasks because we need to produce a label for every token. For work and experiments with models that address only POS tagging, instead of multi-task models for NER and POS tagging, please consult the companion thesis of Chrysa Dikonimaki [Dik21].

3.2 Related Work

Multitask learning (MTL) in NLP has been used for a long time. Even before Deep Learning [Col+11] jointly trained classifiers for NER, POS, Semantic Role Labeling and other tasks and [Mil+00] also jointly trained a NER tagger, a POS tagger and a Constituent parser. [SG16] also use multitask learning for CCG supertagging and POS tagging; for POS tagging the loss is computed from shallower representations in their neural network architecture. More recently, [McC+18] proposed a question-answering framework that deals with 10 different NLP tasks. In this framework, a model is given a question along with a context and then it is asked to find the proper answer by generating text. Finally [Rud17] gives a good overview of multitask learning techniques in Deep Learning. Our work falls in the category of hard parameter sharing as described in [Rud17].

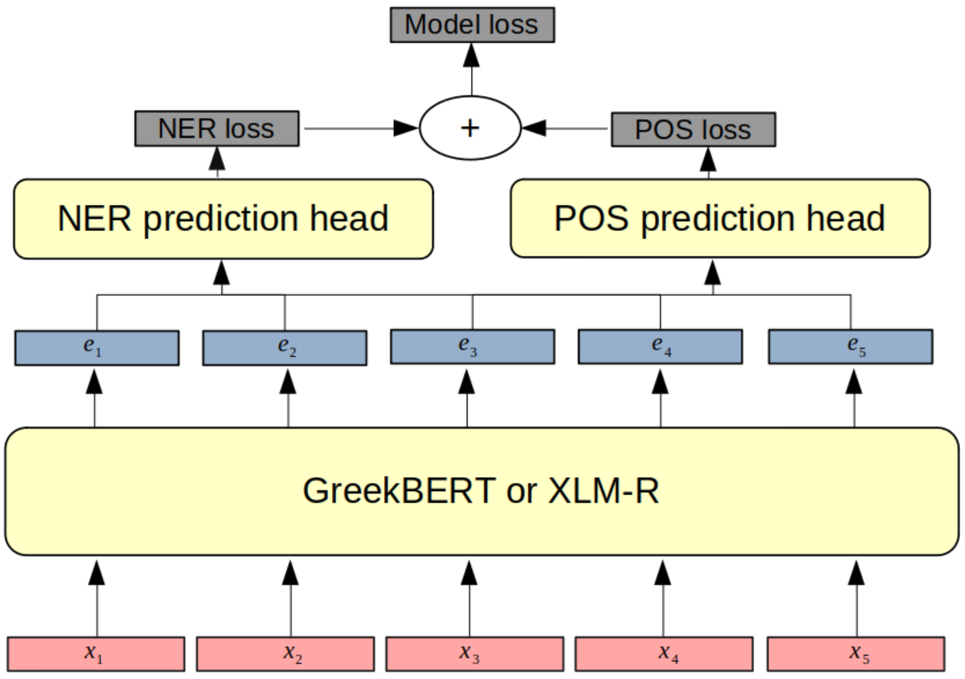
[1]https://spacy.io/
[2]https://www.nltk.org/

3.3 Dataset

The dataset we used for NER is the one presented in [BMK20] and the dataset we used for POS tagging is the Greek treebank from the Universal Dependencies dataset [Niv+16]. For NER we used the inputs and labels as described in section 2.1. For POS tagging we used the UPOS fields which are the universal part-of-speech tag as well as the FEATS fields which contain morphological features for each token.

3.4 Model

Fig. 3.1: The architecture of the MTL model following the hard parameter sharing paradigm.



The model works by sharing a common representation for the two tasks. GreekBERT or XLM-R produces embeddings for every token of an input sentence and then these are fed into the task specific heads.

For the NER prediction head, the BERT embeddings are linearly transformed and then the probabilities for each label of every token are computed with a softmax layer.

Similarly for the POS prediction head, the BERT embeddings are linearly transformed, but with $N + 1$ different linear layers, one for each of the N morphosyntactic categories and another one for the Universal part-of-speech tag. Finally, softmax is applied to each one of these outputs to produce the label probabilities.

More formally for input tokens $x_i, i = 0, \ldots, n$, GreekBERT or XLM-R generates embeddings $e_i, i = 0, \ldots, n$, where for every i, $e_i \in \mathbf{R}^{768}$. For NER, these embeddings are transformed to

$$y_i^{\text{NER}} = A_{\text{NER}} e_i + b_{\text{NER}} \in \mathbf{R}^{C_{\text{NER}}}, i = 0, \ldots, n$$

where C_{NER} is the number of possible classes for each token for the NER task. The predicted classes will be the ones with the greatest score for every sentence. A_{NER} and b_{NER} are parameters of the model and $A_{\text{NER}} \in \mathcal{R}^{C_{\text{NER}} \times 768}$, $b_{\text{NER}} \in \mathcal{R}^{C_{\text{NER}}}$

We denote as $C_{\text{MS}j}$ the number of classes for the morphosyntactic category j and also C_{POS} as the number of classes for the universal part-of-speech tag. The class scores for the morphosyntactic category j are

$$y_i^{\text{MS}_j} = A_{\text{MS}_j} e_i + b_{\text{MS}_j} \in \mathbf{R}^{C_{\text{MS}_j}}, i = 0, \ldots, n$$

A_{MS_j} and b_{MS_j} are parameters of the model and $A_{\text{MS}_j} \in \mathcal{R}^{C_{\text{MS}_j} \times 768}$, $b_{\text{MS}_j} \in \mathcal{R}^{C_{\text{MS}_j}}$

And the class scores for the universal part-of-speech tags are

$$y_i^{\text{POS}} = A_{\text{POS}} e_i + b_{\text{POS}} \in \mathbf{R}^{C_{\text{POS}}}, i = 0, \ldots, n$$

A_{POS} and b_{POS} are parameters of the model and $A_{\text{POS}} \in \mathcal{R}^{C_{\text{POS}} \times 768}$, $b_{\text{POS}} \in \mathcal{R}^{C_{\text{POS}}}$

For each of these output scores we can calculate a cross-entropy loss when we have the true labels available. These losses are then summed which produces a total model loss

$$\mathcal{L}(x_i) = \mathcal{L}_{\text{NER}}(x_i) + \mathcal{L}_{\text{POS}}(x_i) + \sum_j \mathcal{L}_{\text{MS}_j}(x_i)$$

3.5 Experimental Results

3.5.1 Training Details

We tuned the model for the different hyperparameters in 3.1 and optimized with grid search the sum of the macro-f1 score of the UPOS tag and the macro-f1 score of the NER task. The best hyperparameters were 3e-05 for the learning rate, 0.2 for the dropout and 4 for

Learning rate	Dropout	Grad accumulation steps
[5e-5, 3e-5, 2e-5]	[0, 0.1, 0.2]	[4,8]

Tab. 3.1: Hyperparameters that were tested while tuning.

the gradient accumulation steps. During training, batches from each task are interleaved. Whenever a POS tagging batch is selected, the model receives a loss only from the POS prediction head and whenever a NER batch is selected, the model receives a loss only from the NER prediction head. For training we used the AdamW optimizer from PyTorch [LH17] and the cross-entropy loss as the training objective.

Early stopping was also used on the aforementioned sum in order to avoid overfitting. However, we tracked 3 different metrics during early stopping, the macro-f1 score of UPOS tag, the macro-f1 score of the NER classes, the sum of these two scores and kept track of the epoch number and the states of the model where each of these metrics was maximized. We call the checkpoint for the model best at the UPOS metric **MTL@UPOS**, the model best at the NER metric **MTL@NER** and the model best at the sum **MTL@UPOS+NER**.

We chose to optimize the macro-f1 score during hyperparameter tuning because they are less dependant on the distributions of the classes present in the test data and thus they can better reflect the performance of the model on unseen examples and distributions.

The datasets for the two tasks are different, meaning that only one loss is calculated each time, but the sentences of each dataset share words and structure because they are in the same language and therefore information about one task can be propagated to the other task over the training steps. For separate POS models, we use results from the companion thesis of Chrysa Dikonimaki.

3.5.2 Results

We compared the performance of our jointly trained models with the models that were trained for each task separately.

In Table 3.2 the results show that for predicting morphosyntactic categories the models perform similarly. Exceptions are the Abbr and NumType categories where the seperately trained model performed better than both MTL models. Finally we can notice that the information from the NER dataset especially helped it achieve substantially higher macro-f1 scores in the Degree category.

Regarding universal part-of-speech performance in Table 4.4 the MTL@UPOS+NER model performed slightly worse in both macro and micro F1 scores. Multitask learning does seem

Morphosyntactic category	MTL@UPOS+NER		MTL@UPOS		Separate UPOS	
	micro-f1	macro-f1	micro-f1	macro-f1	micro-f1	macro-f1
Case	0.98	0.98	0.98	0.98	0.98	0.97
Definite	1	1	1	1	1	1
Gender	0.98	0.97	0.98	0.97	0.98	0.98
Number	0.99	0.99	0.99	0.99	1	1
PronType	1	0.99	1	0.97	1	0.97
Foreign	0.79	0.79	**0.89**	**0.89**	0.88	0.88
Aspect	0.98	0.98	0.99	0.99	0.99	0.99
Mood	1	**0.97**	1	**0.97**	0.99	0.83
Person	1	0.99	1	1	0.99	0.98
Tense	0.99	0.99	0.99	1	1	1
VerbForm	0.99	0.92	0.99	0.92	0.99	0.93
Voice	0.96	0.96	0.96	0.96	0.96	0.96
NumType	0.94	0.91	0.92	0.85	**0.96**	**0.94**
Poss	0.99	0.99	0.98	0.98	0.98	0.98
Degree	0.92	0.61	**0.96**	**0.86**	0.92	0.49
Abbr	0.88	0.88	0.84	0.84	**0.94**	**0.94**

Tab. 3.2: Micro-f1 and macro-f1 results for the multitask models and the model trained solely on POS tagging.

Metric	MTL@UPOS+NER	MTL@UPOS	Separate UPOS
micro-f1	0.9789	0.9809	**0.9816**
macro-f1	0.9588	**0.9657**	0.9637

Tab. 3.3: F1 results for universal part-of-speech classes.

to help the MTL@UPOS model exceed the initial model in macro-f1 score but not by a large margin.

Finally in NER at Table 3.4 we also see that the multitask models perform similarly with the separately trained model except on the Language class where the MTL@NER underperformed, the Law and Quantity class where the MTL@NER exceeded the other models, the Product class where MTL@UPOS+NER exceeded the other models. Overall the macro and micro F1 scores were very similar.

3.5.3 Discussion

Contrary to some results in the literature, jointly training with these two different tasks only slightly improves the performance if we focus the model to do well on one task. However the model optimized to perform well on both tasks achieves similar results

Named entity	MTL@UPOS+NER	MTL@NER	Separate NER
Cardinal	0.92	0.94	0.94
Date	0.92	0.92	0.93
Event	0.65	0.63	0.64
FAC	0.62	0.70	0.61
GPE	0.93	0.94	0.93
Language	**0.89**	0.73	**0.89**
Law	0.74	**0.78**	0.74
LOC	0.77	0.79	0.80
Money	0.97	0.98	0.98
NORP	0.89	0.90	0.89
Ordinal	0.95	0.95	0.96
Organisation	0.87	0.88	0.88
Percent	0.98	0.98	0.99
Person	0.95	0.96	0.96
Product	**0.81**	0.75	0.75
Quantity	0.85	**0.91**	0.87
Time	0.91	0.92	0.89
Work of art	0.74	0.75	**0.83**
Micro avg	0.90	**0.91**	**0.91**
Macro avg	0.85	**0.86**	**0.86**

Tab. 3.4: F1 scores for each named entity for the jointly trained model and the separately trained model.

with the specialized models. The latter model on the other hand, needs to only call the GreekBERT transformer once and the predictions for NER and POS are produced from the output embeddings. In order for GreekBERT to produce embeddings for one sentence, a large and complex function must be computed which has a long duration. This results in great speedup in situations where we would need both NER tags and POS tags and we would be willing to sacrifice a small part of the model's accuracy on the tasks for speed.

4 Multitask Dependency Parsing - Part of Speech Tagging

4.1 Introduction

Fine grained POS tagging reveals useful syntactic information about every word in a sentence. Very relevant to this task is dependency parsing (DP). Dependency parsing is the task of determining which word depends on which in a sentence as well as the type of this dependency which is called a dependency relation. For example, for POS tagging we need to determine *if* a word is an adverb but in dependency parsing we need to find *which* word is modified by an adverb.

Because these tasks are so closely related, they are often combined. Many dependency parsers use externally provided POS predictions to make better predictions, or use POS predictions from another model in the pipeline and then they are jointly trained.

In our model we combine the two tasks with a Multi Task Learning (MTL) approach to create a tagger and parser. As in Chapter 3, our MTL approach falls in the category of hard parameter sharing as described in [Rud17].

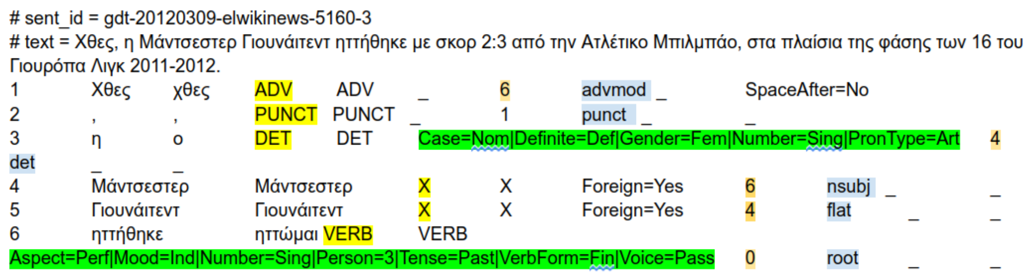
For fine-grained POS tagging and Dependency Parsing the labels are numerous. We refer the reader to Universal Dependencies website, where complete lists of UPOS tag, morphological feature and dependency relation labels exist.[1][2][3]

4.2 Dataset

We used the Greek treebank from the Universal Dependencies dataset [Niv+16] which contains annotated sentences from multiple languages. Every word is annotated with the universal POS tag (UPOS), morhological features (FEATS), the dependant word index (HEAD) as well as the dependency relation (DEPREL).

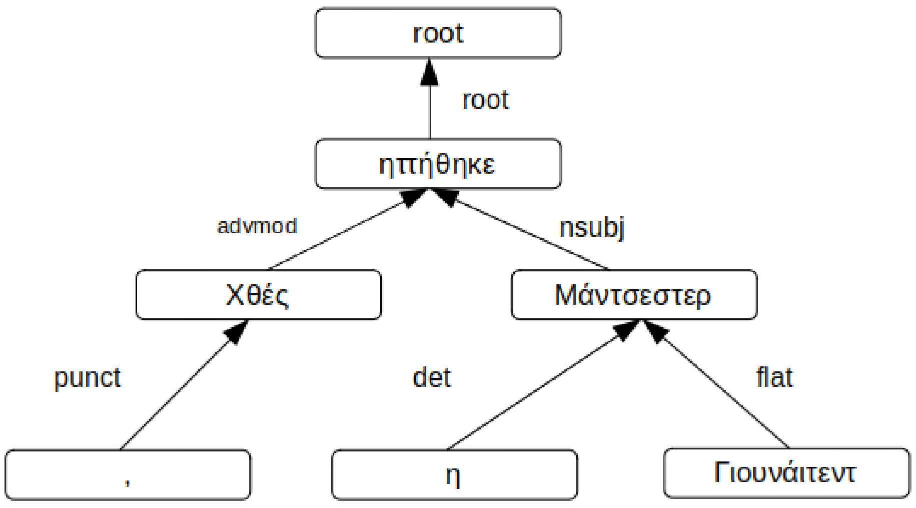
[1]https://universaldependencies.org/u/pos/
[2]https://universaldependencies.org/u/feat/index.html
[3]https://universaldependencies.org/u/dep/

Fig. 4.1: The sentence "Yesterday, Manchester United lost" in CoNLL-U Format. The universal part-of-speech tag is highlighted in yellow, the morphological tags are highlighted in green, the index of the dependant word is highlighted with light orange and finally the dependency relation is highlighted in light blue.



```
# sent_id = gdt-20120309-elwikinews-5160-3
# text = Χθες, η Μάντσεστερ Γιουνάιτεντ ηττήθηκε με σκορ 2:3 από την Ατλέτικο Μπιλμπάο, στα πλαίσια της φάσης των 16 του
Γιουρόπα Λιγκ 2011-2012.
1       Χθες    χθες    ADV     ADV     _       6       advmod  _       SpaceAfter=No
2       ,       ,       PUNCT   PUNCT   _       1       punct   _       _
3       η       ο       DET     DET     Case=Nom|Definite=Def|Gender=Fem|Number=Sing|PronType=Art       4
det     _       _
4       Μάντσεστερ      Μάντσεστερ      X       X       Foreign=Yes     6       nsubj   _       _
5       Γιουνάιτεντ     Γιουνάιτεντ     X       X       Foreign=Yes     4       flat    _       _
6       ηττήθηκε        ηττώμαι VERB    VERB
Aspect=Perf|Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin|Voice=Pass    0       root    _       _
```

Fig. 4.2: The dependency graph of the sentence "Yesterday, Manchester United lost" in CoNLL-U Format. An arc going from word$_i$ to word$_j$ means that word$_i$ depends on (or modifies) word$_j$. For example the word Χθές (Yesterday) is an adverb that modifies ηττήθηκε (lost).



4.3 Model

The model works by sharing a common representation for the two tasks. GreekBERT or XLM-R produces embeddings for every token of an input sentence and then these are fed into the task specific heads.

For the POS prediction head, the BERT embeddings are linearly transformed by $N + 1$ different linear layers, one for each of the N morphosyntactic categories and 1 more for the universal part-of-speech tag.

Sentence	Head	Deprel	POS tag	Features
Χθες	6	advmod	ADV	None
,	1	punct	PUNCT	None
η	4	det	DET	'Case': 'Nom', 'Definite': 'Def', 'Gender': 'Fem', 'Number': 'Sing', 'PronType': 'Art'
Μάντσεστερ	6	nsubj	X	'Foreign': 'Yes'
Γιουνάιτεντ	4	flat	X	'Foreign': 'Yes'
ηττήθηκε	0	root	VERB	'Aspect': 'Perf', 'Mood': 'Ind', 'Number': 'Sing', 'Person': '3', 'Tense': 'Past', 'VerbForm': 'Fin', 'Voice': 'Pass'
με	8	case	ADP	None
σκορ	6	obl	NOUN	'Case': 'Acc', 'Gender': 'Neut', 'Number': 'Plur'
2:3	8	nmod	NUM	'NumType': 'Card'
από	12	case	ADP	None
την	12	det	DET	'Case': 'Acc', 'Definite': 'Def', 'Gender': 'Fem', 'Number': 'Sing', 'PronType': 'Art'
Ατλέτικο	6	obl:agent	X	'Foreign': 'Yes'
Μπιλμπάο	12	flat	X	'Foreign': 'Yes'
,	17	punct	PUNCT	None
στα	None			None
σ	17	case	ADP	None
τα	17	det	DET	'Case': 'Acc', 'Gender': 'Neut', 'Number': 'Plur'
πλαίσια	6	obl	NOUN	'Case': 'Acc', 'Gender': 'Neut', 'Number': 'Plur'
της	19	det	DET	'Case': 'Gen', 'Definite': 'Def', 'Gender': 'Fem', 'Number': 'Sing', 'PronType': 'Art'
φάσης	17	nmod	NOUN	'Case': 'Gen', 'Gender': 'Fem', 'Number': 'Sing'
των	21	det	DET	'Case': 'Gen', 'Definite': 'Def', 'Gender': 'Fem', 'Number': 'Plur', 'PronType': 'Art'
16	19	nmod	NUM	'NumType': 'Card'
του	23	det	DET	'Case': 'Gen', 'Definite': 'Def', 'Gender': 'Neut', 'Number': 'Sing', 'PronType': 'Art'
Γιουρόπα	21	nmod	X	'Foreign': 'Yes'
Λιγκ	23	flat	X	'Foreign': 'Yes'
2011-2012	23	nmod	NUM	'NumType': 'Card'
.	6	punct	PUNCT	None

Tab. 4.1: Sentence example 2.

For the Dependency Parsing prediction head we used the architecture specified in [DQM17]. The difference is that we use GreekBERT's embeddings instead of an LSTM and we don't use POS embeddings.

More formally for input tokens $x_i, i = 0, \dots, n$, GreekBERT or XLM-R generates embeddings $e_i, i = 0, \dots, n$, where for every i, $e_i \in \mathbf{R}^{768}$.

We denote as $C_{\text{MS}j}$ the number of classes for the morphosyntactic category j and also C_{POS} as the number of classes for the universal part-of-speech tag. The class scores for the morphosyntactic category j are

$$y_i^{\text{MS}_j} = A_{\text{MS}_j} e_i + b_{\text{MS}_j} \in \mathbf{R}^{C_{\text{MS}j}}, i = 0, \dots, n$$

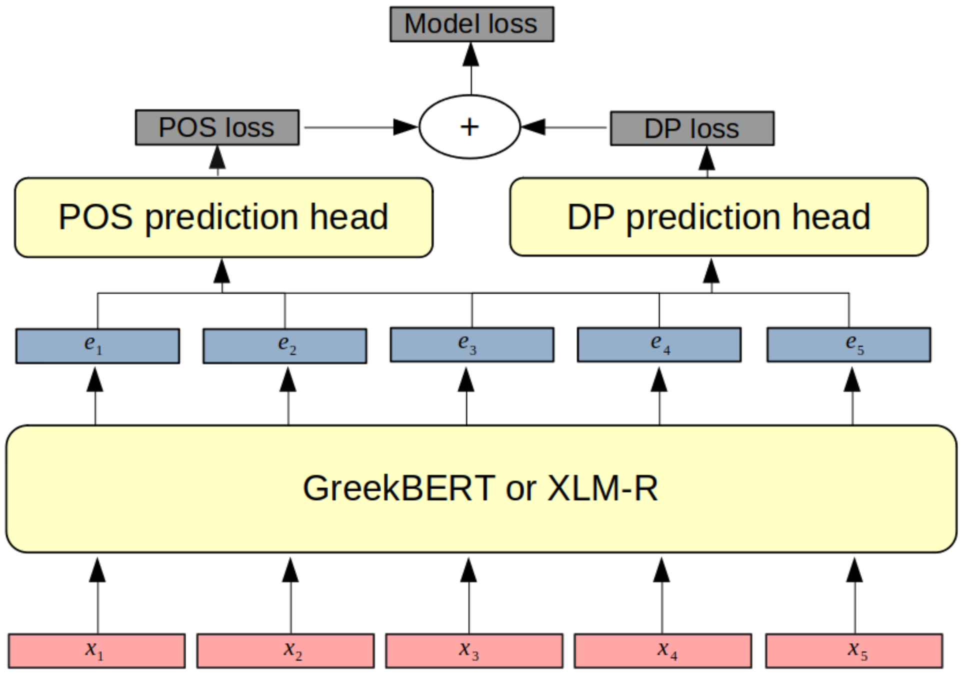
A_{MS_j} and b_{MS_j} are parameters of the model and $A_{\text{MS}_j} \in \mathcal{R}^{C_{\text{MS}_j} \times 768}, b_{\text{MS}_j} \in \mathcal{R}^{C_{\text{MS}_j}}$

And the class scores for the universal part-of-speech tags are

$$y_i^{\text{POS}} = A_{\text{POS}} e_i + b_{\text{POS}} \in \mathbf{R}^{C_{\text{POS}}}, i = 0, \dots, n$$

A_{POS} and b_{POS} are parameters of the model and $A_{\text{POS}} \in \mathcal{R}^{C_{\text{POS}} \times 768}, b_{\text{POS}} \in \mathcal{R}^{C_{\text{POS}}}$

Fig. 4.3: The architecture of the MTL model following the hard parameter sharing paradigm.



For dependency parsing, the following representations are created from the embeddings:

$$\begin{aligned} h_i^{(arc-dep)} &= W^{(arc-dep)} e_i \\ h_i^{(arc-head)} &= W^{(arc-head)} e_i \\ h_i^{(rel-dep)} &= W^{(rel-dep)} e_i \\ h_i^{(rel-head)} &= W^{(rel-head)} e_i \end{aligned}$$

The matrices $W^{(arc-dep)}, W^{(arc-head)}, W^{(rel-dep)}, W^{(rel-head)}$ are parameters of the model which are learned, and their sizes are 768x768 meaning that the produced representations have the same dimensions as the original embeddings.

These representations also have intuitive meanings. $h_i^{(arc-dep)}$ is the representation of token i as a head seeking its dependant and $h_i^{(arc-head)}$ is the representation of token i as a dependant looking for its head.

We then produce scores for each possible arc (i, j):

$$s_{ij}^{(arc)} = (h_j^{(arc-head)})^T W^{(arc)} h_i^{(arc-dep)} + (h_j^{(arc-head)})^T b^{(arc)}$$

Here, $W^{(arc)}$ and $b^{(arc)}$ are parameters that are learned and have size 768x768 and 768 respectively. These terms have intuitive interpretations. The first term is the probability of an arc (i, j) existing given that the dependant is word i and the head word j and the second term is the probability of an arc (i, j) existing only given that the head is word j.

With these scores we can then predict the head for each token x_i as the receiving end of the arc (i, j) with the highest score.

$$y_i^{(arc)} = \arg\max_j s_{ij}^{(arc)}$$

This can also be thought as a sequence-labeling problem because we need to predict one label (the head) for each token.

For the label scores we generate for each arc (i, j), K scores, one for each possible dependency relation. (Note that $\oplus$ means concatenation)

$$s_{ijk}^{(rel)} = (h_j^{(rel-head)})^T U_k^{(rel)} h_i^{(rel-dep)} + w_k^T (h_j^{(rel-head)} \oplus h_i^{(rel-dep)}) + b_k^{(rel)}$$

The intuitive interpretation here is that the first term is the probability of the label being k given that the head is j and the dependant i. The second term is the probability of the label being k given that the head is j, plus the probability of the label being k given that the dependent is i.

The difference comparing to the arc scores is that a linear layer is used for the labels and we need essentially K outputs for the bilinear layer, which is achieved with the K different U_k^{rel} matrices. The parameters here are the K U_k^{rel} matrices each of size 768x768, the K w_k vectors of size 1536x1 and the bias terms $b = b_k^{(rel)}$.

The predictions then for the labels use the predictions for the arcs.

$$y_i^{(rel)} = \arg\max_k s_{iy_i^{(arc)}k}^{(rel)}$$

For each of these output scores we can calculate a cross-entropy loss when we have the true labels available. These losses are then summed which produces a total model loss.

$$\mathcal{L}(x_i) = \mathcal{L}_{\text{ARC}}(x_i) + \mathcal{L}_{\text{REL}}(x_i) + \mathcal{L}_{\text{POS}}(x_i) + \sum_j \mathcal{L}_{\text{MS}_j}(x_i)$$

4.4 Experimental Results

4.4.1 Evaluation

The metrics of interest are Unlabeled Attachment Score (UAS) and Labeled Attachment Score (LAS) for DP. UAS is the number of words that got assigned the correct head divided by the total words and LAS is the number of words that got assigned the correct head and dependency relation to the head divided to the total number of words.

Regarding fine grained POS tagging, we calculate the f1-scores for the UPOS tag and calculate a modified version of f1-score for the morphosyntactic categories. Specifically, for a given gold UPOS tag there is a strict subset of morphological features that can be present and for this reason we ignore the model's outputs if there are any for the morphosyntactic categories that have never been found to occur with the gold UPOS tag.

4.4.2 Training Details

We tuned the hyperparameters once for each metric of focus. We tuned once, optimizing the UAS score, once optimizing the LAS score and once optimizing the macro-f1 score of the UPOS tag. Early stopping was also used in each scenario in order to avoid overfitting. For training we used the AdamW optimizer from PyTorch [LH17] and the cross-entropy loss as the training objective.

The model optimized for UAS had as hyperparameters 5e-05 for the learning rate, 0.2 for the dropout, 4 for the gradient accumulation steps and 0.5 for the multitask loss balance parameter. We call this model **MTL@UAS**. In the LAS case the optimal hyperparameters were also the same. When optimizing for the macro-f1 of the UPOS tag, the optimal hyperparameters were 5e-05 for the learning rate 0.1 for the dropout, 8 for the gradient accumulation steps and 0.8 for the multitask loss balance term (the loss of the POS task was higher). We call the last model **MTL@POS**.

During training batches are formed of multiple sentences from the Universal Dependencies dataset. Every sentence is annotated with both the dependency tree and relations as well as the universal POS tag and morphological features.

Learning rate	Dropout	Grad accumulation steps	λ
[5e-5, 3e-5, 2e-5]	[0, 0.1, 0.2]	[4,8]	[0.2 , 0.5 , 0.8]

Tab. 4.2: Hyperparameters that were tested while tuning.

Morphosyntactic category	MTL@UPOS		Separate UPOS	
	micro-f1	macro-f1	micro-f1	macro-f1
Case	0.98	**0.99**	0.98	0.97
Definite	1	1	1	1
Gender	0.98	0.98	0.98	0.98
Number	0.99	0.99	**1**	**1**
PronType	1	0.95	1	**0.97**
Foreign	0.86	0.86	**0.88**	**0.88**
Aspect	0.99	0.99	0.99	0.99
Mood	0.99	0.66	0.99	**0.83**
Person	1	**1**	1	0.98
Tense	1	1	1	1
VerbForm	0.99	**0.94**	0.99	0.93
Voice	**0.97**	0.96	0.96	0.96
NumType	0.94	0.91	**0.96**	**0.94**
Poss	**0.99**	**0.99**	0.98	0.98
Degree	0.92	0.49	0.92	0.49
Abbr	**0.95**	**0.95**	0.94	0.94

Tab. 4.3: Micro-f1 and macro-f1 results for the multitask model and the model trained solely on POS tagging.

4.4.3 Results

We compared the performance of our jointly trained models with the models that were trained for each task separately. Specifically, we compared the MTL model of POS tagging and DP optimized for UAS, with the model trained only with the DP task in Table 4.5. We also compare the MTL model of POS tagging and DP optimized for the macro-f1 score of the UPOS tag in Tables 4.3 and 4.4

In Table 4.3 the results show that for predicting morphosyntactic categories the models perform very similarly in all categories.

Regarding universal part-of-speech performance in 4.4 the jointly trained model performed slightly better in both metrics but the improvement was marginal.

	MTL@POS	Separate UPOS
micro-f1	**0.9817**	0.9816
macro-f1	**0.9655**	0.9637

Tab. 4.4: F1 results for universal part-of-speech classes.

Metric	MTL@UAS	Separate DP
UAS	0.9208	**0.9357**
LAS	0.9016	**0.9158**

Tab. 4.5: UAS and LAS scores for the MTL model trained on DP and POS as well as the Separate model trained only on DP.

Finally in DP at table 4.5 we see that the optimized MTL model suffers great losses of performance as it scores around 0.015 lower on both UAS and LAS metrics.

4.4.4 Discussion

Once again, the multitask models presented above do not present significant gains for each task. However the multitask models need to only call the GreekBERT model once and the predictions for DP and POS are produced from the same output embeddings. This results in great speedup in situations where we would need both POS tags and the dependency tree of a sentence because GreekBERT needs a lot of computing power to produce the output embeddings.

5 Conclusions and Future work

5.1 Key takeaways

We developed a model that can perform NER, POS and DP in Greek. The separate models for each task have pretty good performances. For NER our best results are 0.92 micro-f1 and 0.87 macro-f1. For POS our best results are 0.98 macro-f1 and 0.97 micro-f1 and for DP our best results are 0.94 UAS and 0.92 LAS. Significant differences in performance were observed in DP, where our model outperformed Stanza's Greek dependency parser by a large margin. XLMR also performed poorly in DP achieving 0.89 and 0.87 UAS and LAS respectively.

Unfortunately, MTL did not enable the models to achieve much better prediction performance in any task, and the MTL models that were optimized to perform equally on their assigned tasks suffered small losses in prediction performance. The benefit of MTL is that, the user can sacrifice a small bit of prediction accuracy to approximately double the prediction generation speed if the user wants both predictions for NER and POS or both predictions for POS and DP.

5.2 Further work

The toolkit of this thesis and the companion thesis pipeline can be extended with many more tasks. First of all, toxicity detection can be added. Toxicity detection is very important for use cases where users can comment or review a certain product. In these cases the inputs from the users need to be moderated in case they contain harmful language [Pav+17], [Pav+20], [Pav+21].

Making the pipeline more efficient is also very important not only for the environmental cost of the models but also for speeding up systems. Distillation [HVD15] and Quantization [Hub+17] are widely used techniques especially in BERT models. A user could theoretically choose some amount of accuracy loss in return for speed. This could further be combined with MTL for more speed up gains.

Additionally, there are many ways to do MTL which were not explored in this work such as those mentioned in Section 3.2. A nice idea would be to use lower level Transformer embeddings for POS tagging while using the final layer Transformer embeddings for Dependency Parsing. Also more combinations of tasks could be explored such as all three of NER, POS and DP together.

Another unexplored idea is adding hyperparameters to the multitask models' loss function to control the effect of the gradients of each task. For example instead of adding the losses in the multitask model of NER and POS we could introduce a hyperparameter β between 0 and 1 so the final loss would be a linear combination of the two losses.

Finally, it's good to have highly tuned models for these tasks because the users of the features of the toolkit will benefit greatly from an increase in performance. There are many parameters that can be tuned more and alternative learning strategies. For example for hyperparameter tuning, Hyperopt [BYC+13] can be used to also utilize possible hyperparameters that are continuous such as the learning rate which we tuned with grid search.

Bibliography

[BMK20] Nikos Bartziokas, Thanassis Mavropoulos, and Constantine Kotropoulos. “Datasets and Performance Metrics for Greek Named Entity Recognition”. In: *11th Hellenic Conference on Artificial Intelligence (SETN 2020).* SETN 2020. Athens, Greece: Association for Computing Machinery, 2020, pp. 160–167.

[BYC+13] James Bergstra, Dan Yamins, David D Cox, et al. “Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms”. In: *Proceedings of the 12th Python in science conference.* Vol. 13. Citeseer. 2013, p. 20.

[Col+11] Ronan Collobert, Jason Weston, Léon Bottou, et al. “Natural language processing (almost) from scratch”. In: *Journal of machine learning research* 12.ARTICLE (2011), pp. 2493–2537.

[Con+19] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, et al. “Unsupervised cross-lingual representation learning at scale”. In: *arXiv preprint arXiv:1911.02116* (2019).

[Dev+18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[Dik21] Chrysa Dikonimaki. “A Transformer-based Natural Language Processing Toolkit – Part-of-Speech Tagging and Dependency Parsing”. 2021.

[DQM17] Timothy Dozat, Peng Qi, and Christopher D Manning. “Stanford’s graph-based neural dependency parser at the conll 2017 shared task”. In: *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies.* 2017, pp. 20–30.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Hub+17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. “Quantized neural networks: Training neural networks with low precision weights and activations”. In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 6869–6898.

[HVD15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[Jur20] Dan Jurafsky. *Speech & language processing*. 2020.

[Kou+20] John Koutsikakis, Ilias Chalkidis, Prodromos Malakasiotis, and Ion Androutsopoulos. “Greek-bert: The greeks visiting sesame street”. In: *11th Hellenic Conference on Artificial Intelligence*. 2020, pp. 110–117.

[Lam+16] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. “Neural architectures for named entity recognition”. In: *arXiv preprint arXiv:1603.01360* (2016).

[LH17] Ilya Loshchilov and Frank Hutter. “Decoupled weight decay regularization”. In: *arXiv preprint arXiv:1711.05101* (2017).

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: (2001).

[MAM08] Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. “Named entity recognition approaches”. In: *International Journal of Computer Science and Network Security* 8.2 (2008), pp. 339–344.

[McC+18] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. “The natural language decathlon: Multitask learning as question answering”. In: *arXiv preprint arXiv:1806.08730* (2018).

[Mil+00] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. “A novel use of statistical parsing to extract information from text”. In: *1st Meeting of the North American Chapter of the Association for Computational Linguistics*. 2000.

[Niv+16] Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, et al. “Universal dependencies v1: A multilingual treebank collection”. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*. 2016, pp. 1659–1666.

[Pav+17] John Pavlopoulos, Prodromos Malakasiotis, Juli Bakagianni, and Ion Androutsopoulos. “Improved abusive comment moderation with user embeddings”. In: *arXiv preprint arXiv:1708.03699* (2017).

[Pav+20] John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. “Toxicity Detection: Does Context Really Matter?” In: *arXiv preprint arXiv:2006.00998* (2020).

[Pav+21] John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. “Semeval-2021 task 5: Toxic spans detection”. In: *Proceedings of SemEval* (2021).

[Qi+20] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. “Stanza: A Python natural language processing toolkit for many human languages”. In: *arXiv preprint arXiv:2003.07082* (2020).

[RCE+11] Alan Ritter, Sam Clark, Oren Etzioni, et al. "Named entity recognition in tweets: an experimental study". In: *Proceedings of the 2011 conference on empirical methods in natural language processing*. 2011, pp. 1524–1534.

[RM95] Lance A. Ramshaw and Mitchell P. Marcus. "Text Chunking using Transformation-Based Learning". In: *CoRR* cmp-lg/9505040 (1995).

[Rud17] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[SG16] Anders Søgaard and Yoav Goldberg. "rt". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2016, pp. 231–235.

[SSH19] Jana Straková, Milan Straka, and Jan Hajic. "Neural Architectures for Nested NER through Linearization". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5326–5331.

[Tos+15] Melanie Tosik, Carsten Lygteskov Hansen, Gerard Goossen, and Mihai Rotaru. "Word embeddings vs word types for sequence labeling: the curious case of cv parsing". In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 2015, pp. 123–128.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[Yam+20] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. "LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 6442–6454.

List of Acronyms

NLP	Natural Language Processing
NER	Named Entity Recognition
POS	Part of speech tagging
DP	Dependency parsing
MTL	MultiTask Learning
MLM	Masked Language Modelling
NSP	Next Sentence Prediction

List of Figures

List of Tables