Files and Directories

- **/orginal_bert_repo/** is a copy of Greek-Bert's repo.
- **/bert-base-greek-uncased** is Greek-Bert's json idex and vocab
- **/APT/crete_APT_input_files_before_sentence_seperation.ipynb** splits RAPTARCHIS dataset into sentences and normalizes.
- **/APT/TAPT_MBert.ipynb** NOT YET COMPLETE.runs TAPT
- **/dataset/** contains the datasets and vocabularies
- **/models/** contains the models in .pth format for each task. For example /models/greek-BERT/volume_4.pth is a Greek-Bert model fine tuned on volume classification from RAPTARCHIS dataset.
- **/sourses/** is a directory containing papesr, blogs, videos etc that i have read so far (there are more. pm me if you want).
- **/code/data_overlap.ipynb** creates dataset overlap table, exactly like in the paper. The code is copied from this github, with slight modifications. This script was run locally.
- **/code/dataset_overlap_exploration** explores overlaps.pkl from dont_stop_pretraining
- **/code/fragmentation_ratio.ipynb** calculates the fragmentation ratio of multiple models in multiple datasets. This script was run locally.
- **/code/Greek_bert.ipynb** tries to duplicate the experiments done with Greek-Bert in 'Legal Text Classification for Greek Legislation'. Fine-tunes and tests the model on RAPTARCHIS dataset. More details here.
- **/code/Greek_bert_random_numbers.ipynb** is the exact same as Greek_bert.ipynb but with the numbers in the dataset replaced with random numbers
- **/code/Greek_bert_replace_numbers.ipynb** is the exact same as Greek_bert.ipynb but with the numbers in the dataset replaced with #
- **/code/no_numbers_Greek_bert.ipynb** is the exact same as Greek_bert.ipynb but with the numbers in the dataset removed completely
- **/code/Legal_bert.ipynb** is the exact same as Greek_bert.ipynb but using the Greek-Legal-Bert model
- **/code/Legal_bert.ipynb** is the exact same as Greek_bert.ipynb but using the Multilingual-Bert model
- **/code/normalize_dataset_before_sentence_seperation.ipynb** transforms Legal-Bert's, Greek-Bert's and RAPTARCHIS dataset, exactly like in the paper, before sentence separation. The code is copied from this github, with slight modifications. This script was run locally.
- **/code/normalize_dataset_after_sentence_seperation.ipynb** transforms Legal-Bert's dataset, like in the Greek-Bert paper. The transformation is done after sentence separation with The code is copied from this github, with slight modifications. This script was run locally.
- **/code/other_stats.ipynb** calculates how many english and numbers are in each dataset and what year are the laws written in
- **/code/raptarchis_to_pkl.ipynb** transforms RAPTARCHIS dataset from json format to dataframes and saves them to pickles in /dataset/*.pkl

- **/code/vocab_overlap.ipynb** creates vocabulary overlap table. This script was run locally.

Greek-Bert Fine-tuning

paper : ‘Legal Text Classification for Greek Legislation’

Parameters

- **model input:** header + articles, padded to max_batch_size, truncated to 512
- **learning rate:** 0.00002
- **batch size:** 8 (P100 can maybe handle 16)
- **epochs:** 5->volume, 9->chapter, 14->subject. did not continue fine-tuning since i reached the paper’s results.
- **time per epoch:** ~25min->P100, ~52min->T4, >60min->K80

Implementation differences

- I use scheduler `optim.lr_scheduler.ReduceLROnPlateau(mode='min', factor=0.5, patience=1)`
- I stop fine-tuning after reaching the desired accuracy. They use early stopping.
- I don’t use gradient clipping. They use clipvalue=5.0

Results

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	88.36	88.36	88.36	88.36	88.36	88.36	-	-	-
Chapter	83.27	83.27	83.27	83.61	83.75	83.68	52.63	98.04	68.49
Subject	79.27	79.27	79.27	84.43	86.32	85.36	62.84	84.1	71.93

Cleaning of RAPTARCHIS dataset

I noticed a lot of numeric values in the RAPTARCHIS dataset that, I thought, won't help the model predict the correct class. With Christos’s suggestion, I decided to train Greek-Bert on the dataset after removing all the numbers using the following procedure. I just deleted the numeric values, this is the simplest way possible and it definitely isn't perfect ex the sentence “Σύμφωνα με τον νόμο 1234/2006” will become “Σύμφωνα με τον νόμο /”, the ‘/’

will not be deleted. Even this simple cleaning method is enough to help our model learn better, here are the results of ‘chapter’ level classification (389 classes) using Greek-Bert[1]:

	Original dataset	Cleaned dataset
number of epochs / accuracy	9 / 81.4	4 / 81.1

The accuracy of the model trained on the original dataset and tested on a cleaned test set is 82.1. The accuracy of the model trained on the cleaned dataset and tested on the original test set is 80.05.

We clearly observe from the table that the cleaned dataset helped the model reach the same result, needing only half the epochs[2]. We also observe that a cleaned dataset helps the model, trained on the original dataset, make better predictions on the test set.

Is it worth expanding-altering the vocabulary? initial thoughts

I found a master thesis named Effects of Inserting Domain Vocabulary and Fine-tuning BERT for German Legal Language, that essentially tries to do the exact same task with us, as the title suggests. They expect results similar to SciBert +0.6% on F1 but find no significant improvements when comparing the German-Bert with the expanded vocabulary German-Bert. They suggest there is no improvement because “the original vocabulary already includes 90% of the legal vocabulary. This is one of the characteristics that renders the legal domain especially hard to grasp for both humans and machines, many of the day-to-day words are imbued with a different meaning when used in the legal context. SciBERT, the pre-trained BERT model for scientific texts [3], improves around 0.60% on F1 across all the datasets but their original-scientific vocabulary overlap is just 47%.”. I believe this is not the case with the greek language as many of the laws are outdated and written in …………… , but i will have to test my hypothesis. I will explain their technique for expanding the vocabulary in the next section.

There are not many papers that try to adapt Bert’s vocabulary to be more domain specific, except BioBert. Most authors chose to pretrain a Bert from scratch with a new domain specific vocabulary, like Greek-Legal-Bert. There are, however, papers that try to modify multilingual Bert’s vocabulary to create a version of Bert more suitable for a certain low resource language, and report good results. We can assume that for a multilingual Bert, a specific low recourse language’s vocabulary is considered domain-specific vocabulary. It is obvious that, under that assumption, we can use the same techniques to modify Greek-Bert’s vocabulary to the legal domain.

1 on the original training dataset the fragmentation rate is Greek-Bert: 1.645, M-Bert: 2.968
2 I think that the model will barely fit in a V100. If we have the resources for ex-Bert, we could train a Bert-Large instead, using this. That is probably going to have much better results.

Altering-Expanding Greek-Bert's vocabulary

In all of the papers that try to alter the original Bert vocabulary to better suit a specific domain (except exBert's), the original vocabulary size remains the same. The new words are added with the following two ways:

- Unused tokens
  Bert's vocabulary has 1000 [UNUSEDxxx] tokens that are meant to help us expand it without getting into the trouble of changing its size, as that would also mean changing the model's weight dimensions. Greek-Bert only has 59 of those tokens.
- Removing 'unpopular' tokens
  Bert's vocabulary usually has words that do not appear in a specific domain. In our case, we can run the Greek-Bert tokenizer over the whole dataset and flag the tokens that are not used even once. Those tokens exist for sure. For instance, with a quique revision of Greek-Bert vocabulary, it contains a lot of english words and abbreviations that are certainly absent from greek legal documents.

The newly added tokens won't be recognized by Bert, as they won't have trained embeddings. There are different techniques that appear in the literature, for initializing and training those embeddings. I will present an overview of some of those below.

BioBert

In the paper they claim that BioBert has a modified vocabulary to better suit the medical domain. However I found that this is only true for BioBert-large v1.1 and not BioBert-base v1.1 or v1.0 which use the original bert vocabulary (source & source). The paper does not go into detail on how the new word embeddings were initialized (so i assume it was randomly). All in all, they just trained a new tokenizer on medical text, selected the N most used tokens and added them to Bert's original vocabulary. After that, they pretrained their model on biomedical corpuses and the original Bert corpuses (this resembles DAPT).

German-Bert

In this master's thesis they try to expand the German-Bert vocabulary to include legal domain tokens using the same technique SciBert and Bert did, WordPiece. They end up only adding around 3000 tokens to the existing vocabulary and not removing a single token. As they say there was a 90% overlap of the original and legal domain vocabulary. After that they pretrain the embeddings ????????????? finetune the embeddings ??????????????????

Patch-BERT

Patch-Bert introduces several ways to expand or alter Bert's vocabulary. The best results reported are +<1.2%. The methods described in the paper are ways to introduce new subwords in the vocabulary. I will briefly describe some of the methods presented in the paper:

- Find the closest subword, using character distance, to the newly added subword. They share weights.
- Find a subword that is not used in the downstream task and replace it with a new one. The new subword weights are initialized that way (but change during fine-tuning).
- Add a new subword and initialize the weights randomly.
- Find a similar subword, to the new subword, using Bert's mask word prediction! The new subword can share weights or have the same initial weights.

I did not get into details for which methods were used to expand the vocabulary and which to alter it, as i believe all can be used for both. I also think that some of these methods can be used for adding-modifying whole words and not just subwords.

exBert

In this paper they introduce exBert, a modified version of Bert to support an extension of plus 18k tokens to the original vocabulary. In contrast to SciBert they don't just pretrain from scratch a new Bert with a new, bigger, domain specific vocabulary. Their goal is to change the vocabulary size without also changing the model's weight dimensions, as this would mean that the entire model will have to be pretrained from scratch.

To achieve that, they add an 'expansion module' to the vocabulary-to-embedding layer and each hidden layer. The 'expansion module', added to each hidden Bert layer, uses the same transformer-based architecture as Bert, with much smaller sizes. The 'vocabulary-to-embedding expansion' is exactly the same as the original vocabulary-to-embedding layer, with the sole purpose of supporting the new vocabulary tokens. For a better understanding of the architecture see Figure 1 below. During pretraining they froze all the original Bert weights and just trained the 'expansion modules'. They observe a +1-3% F1 score on NER against BioBert. While exBert has more parameters (153M compared to 110M of BioBert), when comparing its performance for the same time of pretraining, exBert outperforms BioBert (note that in the same time of pretraining, for example 24hrs, exBert proceeds through less data when compared to BioBert 24% vs 34%, but it still manages to outperform it).

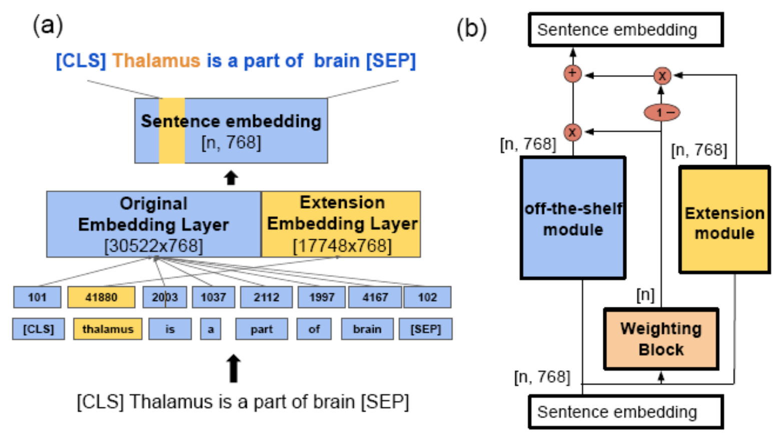


Figure 1: Sentence embedding and the exBERT architecture. (a) Derivation of the sentence embedding based on both the original and extension vocabulary. (b) Each input sentence consists of n 768-dimensional embedding vectors where n is 128 in our experiments. The output embedding is a component-wise weighted (computed by the weighting block) sum of outputs from the two modules.

Distilling knowledge from Greek-Legal-Bert

We could use the technique from Distilling Task-Specific Knowledge from BERT into Simple Neural Networks (as suggested here) or TinyBERT: Distilling BERT for Natural Language Understanding. In these papers they distill knowledge from BERT into smaller models. This is done by using Bert as a ‘teacher’ and a smaller model as a student that tries to imitate Bert’s outputs. We could maybe use this technique to transfer knowledge from Greek-Legal-Bert’s layers to Greek-Bert, but this defeats the purpose of ending up with a smaller model. I do not consider this technique to be what we are looking for, as it essentially suggests training a new model to match the performance and vocabulary of a ‘Master’ model with sota results. We don't yet know if Greek-Legal-Bert outperforms Greek-Bert in legal domain tasks. If not this could lower the performance of Greek-Bert. For that reason I reject this technique for our purposes.

Conclusion

In my opinion the most elegant method is Patch-Bert. exBert seems to have better results than Patch-Bert (1-3% vs <1.2%), although they are not on the same experiments. Both papers try to expand on the same paper, offering different solutions. It is worth noting that both papers were published at the same conference, the same month (November 16–20, 2020).

I believe ex-Bert has better results not only because it expands the vocabulary, but primarily because it adds small layers and more parameters to the preexisting model. I think this method is too complex and is probably not what we had in mind when vocabulary expansion was proposed.

I don't think vocabulary expansion alone is worth it, as it suggests changing the models size and pretrain it (sometimes from scratch). Vocabulary alteration, on the other hand, requires much less computational resources but offers little to no improvement.

What about M-Bert ?

M-Bert (uncased) is the multilingual version of Bert. I will argue that it probably is worth it to expand M-Bert's vocabulary, just like Greek-Bert's vocabulary, and test it on the RAPTARCHIS dataset. I expect M-Bert to yield the best results between our models.

I fine-tuned M-Bert for legal document classification using the cleaned RAPTARCHIS dataset, with the same hyper-parameters used to fine tune Greek-Bert. After 9 epochs it reached an accuracy of 81.04, the same as Greek-Bert. It is worth noting that after the 9th epoch the model overfitted. In comparison Greek-Bert riched the same accuracy on the cleaned dataset after 4 epochs. Those results are consistent with the paper 'Legal Text Classification for Greek Legislation'.

In Greek-Bert's paper the average difference between Greek-Bert's and M-Bert's (uncased) fragmentation rate on the 2 easy (less language specific) tasks (PoS, NER) is ~1.04%. On the difficult NLI (more language specific) it is ~6%. The difference in fragmentation ratio between Greek-Bert and M-Bert in all tasks is ~1.

In our case Greek-Bert's fragmentation rate on the cleaned RAPTARCHIS training dataset is 1.561 and M-Bert's is[3] 2.932. The difference is ~1.4. But as we expect, this does not affect the results, in most cases M-Bert is better than Greek-Bert.

I believe this is because the legal text classification problem is easy, at least not as difficult as NLI. Greek-Bert has definitely a better understanding of the greek language, as shown with it's NLI results. In our easy problem, thought, this deep of an understanding is possibly not needed. M-Bert's significantly larger dataset has given it an edge in understanding language in general and, I believe, this is why it scores better on RAPTARCHIS.

As we saw, M-Bert's knowledge from it's significantly larger dataset overshadows the difficulties that come with the large fragmentation ratio. I believe that greek-Bert's scores will not improve significantly after the vocabulary expansion but M-Bert's will.

Vocabulary overlap

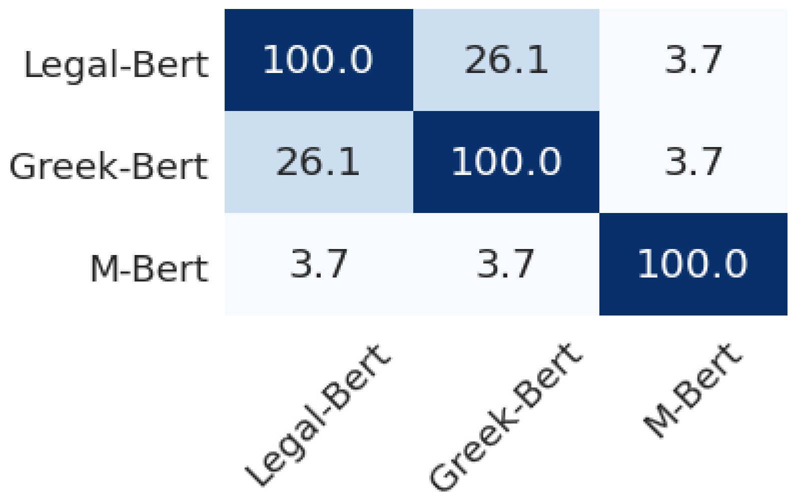
We will measure:

- the overlap between Greek-Bert vocabulary and Greek-Legal-Bert vocabulary
- the overlap between Greek-Bert and Greek-Legal-Bert dataset's
- Fragmentation ratio of Greek-Bert and Greek-Legal-Bert on RAPTARCHIS dataset

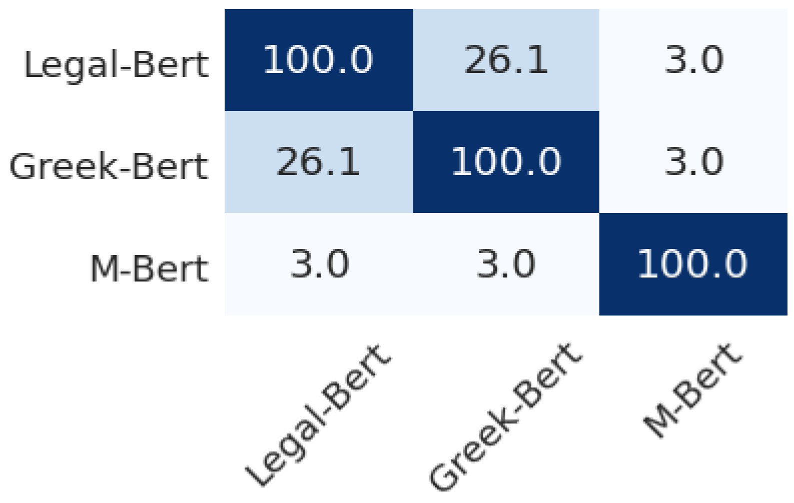
3 I chose to test chapter because it's the middle ground between volume 47 classes and subject 2285 classes, but more tests will be done if deemed worthwhile

Bert vocabulary

considering all M-Bert vocab



considering only greek words from M-Bert vocab

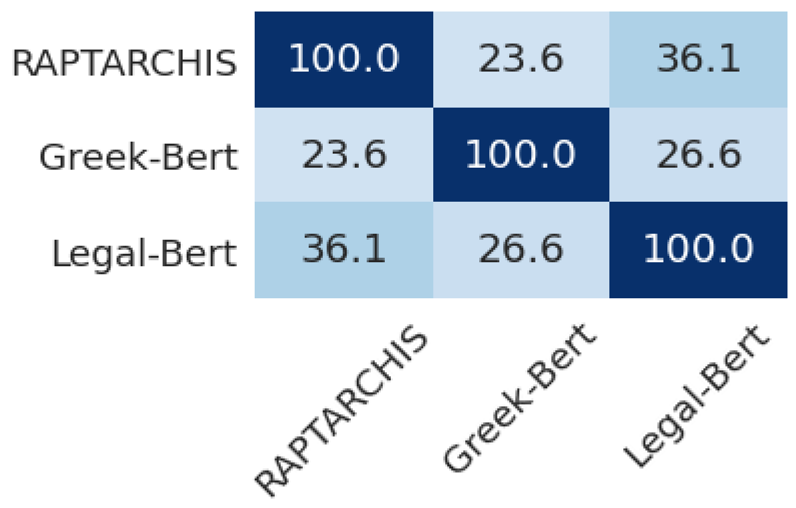


PreTraining datasets

Greek-Bert was trained on the Greek part of Wikipedia, the Greek part of the European Parliament Proceedings Parallel Corpus (Europarl) and the Greek part of OSCAR, a clean version of Common Crawl. Using the code provided in the paper's repo (build_data_el.py and normalize_data.py), we transformed the entire dataset to the desired format. Note: I can not find the exact version of wikipedia Greek-Bert was trained on. I chose this version, it should be close to the version used, considering the date the paper was released. Wikipedia pages don't change that much, so this decision will not affect the results at all.

Greek-Legal-Bert was trained on the Nomothesia dataset. We will transform Greek-Legal-Bert's dataset using the same code as before, so that both datasets are in the same format.

!!!!!!!!!!!!!!!!!! it is very sensitive to limit_min



Fragmentation ratio

We will calculate the fragmentation ratio of Greek-Legal-Bert and Greek-Bert on:

- RAPTARCHIS dataset (training set)
- Nomothesia dataset (one of each source)
- Greek-Bert dataset (Europarl + OSCAR + Greek wiki)

The word fragmentation ratio is defined as the average number of sub-word tokens per word.

	RAPTARCHIS	Nomothesia	Europarl + OSCAR + Greek wiki
Greek-Bert	1.646	1.154	1.151
Legal-Bert	1.513	1.06	1.277
M-Bert	2.968	2.059	2.176

DAPT and TAPT

This paper proposes that we keep pretraining Bert on domain specific unlabeled data (domain adaptive pretraining DAPT) and task specific data (task adaptive pretraining TAPT) and reports a ~0.6-12% score increase.

APT vs Vocabulary alteration-expansion

Vocabulary expansion requires pretraining (sometimes from scratch), and resembles APT a lot. I believe that a vocabulary expansion, like Bio-Bert's, is not worth the trouble, especially when compared to APT. ex-Bert's vocabulary expansion is computationally expensive and somewhat complicated.

From here and here it is obvious that APT yields better results when compared to vocabulary alteration. Vocabulary alteration only helps with lowering the fragmentation ratio. As observed by the findings of Greek-Bert's and Bio-Bert's paper, the fragmentation ratio (and the domain specific words in the vocabulary) is not as important as the pretraining dataset size (and origin, domain specific or not), at least for easy tasks.

Considering all that, we can choose one of the following options:

- APT alone (both DAPT and TAPT as it has better results)
- ex-Bert's approach
- APT with vocabulary alteration, to get the best of both worlds

I would choose option 1 or option 3, even though i don't believe 3's results will be much better than option 1. I reject option 2 as it essentially uses a new model architecture and not the original Bert and it's very computationally expensive[4].

Τρι, Απρίλιος 20

Complete tests with cleaned dataset

As suggested here, I cleaned the RAPTARCHIS dataset, fine-tuned and tested Greek-Bert on it with two different ways.

1. I replaced all the numbers in the dataset with the char '#'. For example "20. συμφωνα με τον νόμο 1234/2020 παρ15" became "#. συμφωνα με τον νόμο #/# παρ#". I replaced the char '5' with '#' in the vocabulary.
2. I completely removed the numbers from the dataset. For example "20. συμφωνα με τον νόμο 1234/2020 παρ15" became ". συμφωνα με τον νόμο / παρ".

Cleaned testing data

The number of epochs need for the following results are:

- no numbers:
  - volume 6
  - chapter 4
  - subject 10
- replace numbers with #
  - volume 4

4 I stopped training after it reached the same accuracy, i don't know if this is the cap

  - chapter 6
  - subject 9

While both techniques seem to need a similar number of epochs, it is worth noting that without numbers the model reached a very high accuracy in about half the total epochs and then gained <+1,2% accuracy increase per epoch. Also, without numbers, the sentence length was reduced and consequently so was the time per epoch.

Numbers replaced

Greek-Bert fine tuned on RAPTARCHIS with all numbers replaced with # symbol. The # symbol replaced the character 5 in bert's vocabulary. Tested with numbers replaced by # in the testing data.

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	88.17	88.17	88.17	88.17	88.17	88.17	-	-	-
Chapter	82.96	82.96	82.96	83.35	83.41	83.38	47.37	91.84	62.5
Subject	74.91	74.91	74.91	80.8	83.12	81.94	55.01	85.91	67.08

No numbers

Greek-Bert fine tuned on RAPTARCHIS with all numbers removed. Tested with numbers removed from testing data.

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	89.4	89.4	89.4	89.4	89.4	89.4	-	-	-
Chapter	81.33	81.33	81.33	82.0	82.03	82.02	17.89	94.44	30.09
Subject	77.09	77.09	77.09	81.24	84.56	82.86	64.26	89.33	74.75

Uncleaned testing data

this is probably useless.

Numbers replaced

Greek-Bert fine tuned on RAPTARCHIS with all numbers replaced with # symbol. The # symbol replaced the character 5 in bert's vocabulary. Tested with numbers intact in the testing data.

	All labels	**Frequent**	**Few shot**

	R	P	F1	R	P	F1	R	P	F1
Volume	88.06	88.06	88.06	88.06	88.06	88.06	-	-	-
Chapter	82.73	82.73	82.73	83.12	83.17	83.14	47.37	91.84	62.5
Subject	74.24	74.24	74.24	80.42	82.87	81.63	53.05	85.32	65.42

No numbers

Greek-Bert fine tuned on RAPTARCHIS with all numbers removed. Tested with numbers intact in the testing data

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	89.0	89.0	89.0	89.0	89.0	89.0	-	-	-
Chapter	80.88	80.88	80.88	81.52	81.56	81.54	20.0	95.0	33.04
Subject	75.94	75.94	75.94	80.08	83.84	81.92	63.08	88.94	73.81

Conclusions

In the volume and chapter classes, removing the numbers from the dataset was clearly a better solution, as we reached the same accuracy with almost half as many epochs. When considering the subject classes, none of the cleaning methods managed to reach the desired accuracy.

The subject classes is the most difficult classification problem, out of the three. Not only does it have a greater number of classes, but it also has a high percentage of few and zero shot classes. Weirdly enough, the model trained without numbers managed to outperform both the original and the one trained with ‘#’ in few-shot predictions, but still fell short in the overall score.

My initial assumption that the numbers are completely irrelevant was obviously incorrect. After some thought I came up with two possibilities that can explain those results.

1. The numbers in the dataset provide no information at all, but act as noise that helps our model generalize. This is not likely because if it were true, the no numbers model wouldn't outperform the ‘#’ one.
2. The numbers in the dataset provide information on the class. For this to happen, some numbers must appear in the dataset a considerable amount of times.

To test the first possibility I fine-tuned a Greek-Bert model on RAPTARCHIS and replaced all the numbers in the dataset with a random number of the same length. The model reached 77.76% after epoch number 11.

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Subject	77.76	77.76	77.76	82.26	84.71	83.46	63.94	85.74	73.25

That shows that the numbers must provide some information. This can only happen if some numbers are repeated a significant amount of times in the dataset. As we can see from the dont_stop_pretraining/overlaps_*.pkl files, a lot of numbers (~350) are repeated more than 100 times in the RAPTARCHIS dataset. For instance, 1948, 1949, 195, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 000, 01, 02, 08, 09, 10, 100, 115, 116, 117, 118, 119, 12, 120, 121.

Dataset overlap sensitivity

As said here, the dataset overlap is very sensitive on the min parameter of CountVectorizer. On the following table are the vocabulary sizes per dataset (Legal-Bert, Greek-Bert, RAPTARCHIS). With # is the vocabulary extracted from the original dataset, with -# is the vocabulary extracted from the dataset after removing all the numbers.

min param	**Legal-Bert**		**Greek-Bert**		**RAPTARCHIS**	
	#	**-#**	**#**	**-#**	**#**	**-#**
4	66773	66047	55218	53852	36350	32814
100	9370	8176	4271	4186	2946	2628
400	3338	2983	1035	991	576	477

Bert vocabulary exploration

In the following table are the number of english words and numbers in Greek and Legal Bert vocabulary.

	english words	**numbers**
Greek-Bert	3847 (11,02%)	627 (1,79%)
Legal-Bert	3000 (8,57%)	3363 (9.6%)

We can see that Greek-Bert has more english words in it's vocabulary. Legal-Bert's vocabulary has more english words than expected. Legal-Bert's vocabulary contains more than 5 times more numbers.

RAPTARCHIS dataset exploration

The language of Greek laws changed from καθαρεύουσα to δημοτική in 1977. As stated here “Ιστορική είναι και η Εγκύκλιος του Γεωργίου Ράλλη για τη χρήση της δημοτικής στη δημόσια διοίκηση, που καθιερώνει τη δημοτική στη δημόσια διοίκηση από 01-02-1977” (other sources 1 and 2).

In the RAPTARCHIS dataset 27807 laws were written before 1977 and 19756 laws after 1977. So around 27807 laws are in καθαρεύουσα.

Greek with transfer learning

M-Bert achieves remarkable results considering its small greek vocabulary and training dataset. This can be attributed to the fact that it was trained on a huge multilingual dataset. That multilingual dataset contained languages that significantly help the greek language learning process. They are, in a sense, similar to greek. Which languages are those?

As stated here, the greek language is part of a larger family named Indo-European, this contains Albanian, Armenian, Balto-Slavic, Celtic, Germanic, Hellenic, Indo-Iranian and Italic. Theoretically, those languages could help our model learn greek easier. As stated here, the greek language is also part of a smaller group named Hellenic languages, but the only modern language of that group is modern greek.

As discussed in this paper Zero-Shot Cross-Lingual Transfer with Meta Learning, hindi, russian, urdu, bulgarian, arabic help Bert learn the greek language faster. (this also could help). Here is a table from the paper, greek is on the 4rth row annotated ‘el’. We observe that all languages tested, except english, help more or less.

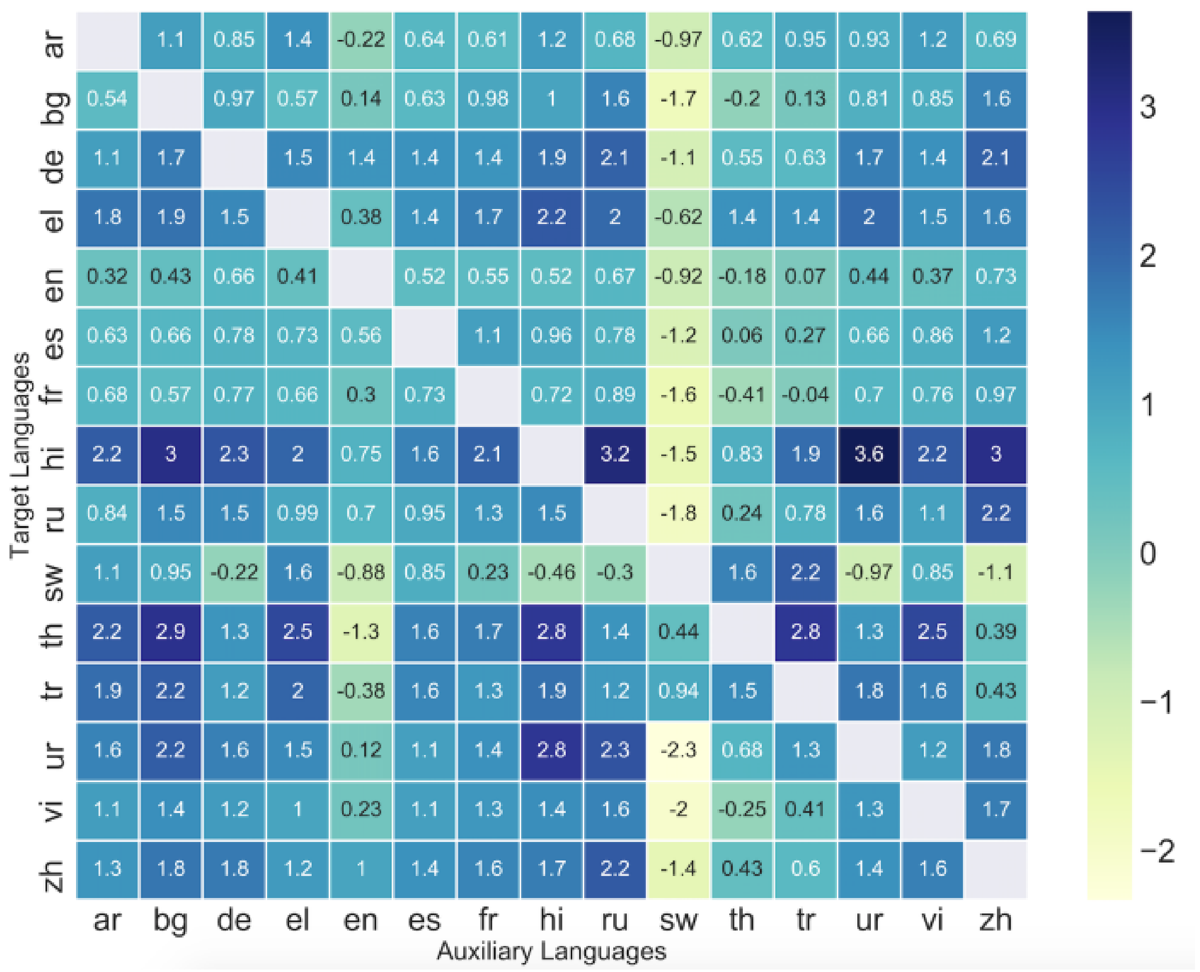


Figure 1: Differences in performance in terms of accuracy scores on the test set for zero-shot X-MAML on XNLI using the Multi-BERT model. Rows correspond to target and columns to auxiliary languages used in X-MAML. Numbers on the off-diagonal indicate performance differences between X-MAML and the baseline model in the same row. The coloring scheme indicates the differences in performance (e.g., blue for large improvement).

M-Bert was trained on all the previously mentioned, as beneficial to the greek, languages. All the languages M-Bert was trained on are here.

Legal-Bert Fine-tuning

The same parameters used for Greek-Bert finetuning are used for Legal-Bert finetuning as well as. The total number of epochs is as follows:

4->volume
7->chapter

14->subject

It is worth noting that the accuracy was really high from the first few epochs. The last epochs only changed the accuracy by little.

Results

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	89.07	89.07	89.07	89.07	89.07	89.07	-	-	-
Chapter	83.98	83.98	83.98	84.42	84.57	84.49	44.21	95.45	60.43
Subject	80.04	80.04	80.04	84.5	86.93	85.7	66.48	82.9	73.79

M-Bert Fine-tuning

The same parameters used for Greek-Bert finetuning are used for M-Bert finetuning as well as. The total number of epochs is as follows:

7->volume
10->chapter
16->subject

Results

	All labels			**Frequent**			**Few shot**		
	R	**P**	**F1**	**R**	**P**	**F1**	**R**	**P**	**F1**
Volume	87.63	87.63	87.63	87.63	87.63	87.63	-	-	-
Chapter	80.77	80.77	80.77	81.15	81.23	81.19	46.32	95.65	62.41
Subject	75.08	75.08	75.08	83.12	84.96	84.03	46.45	71.92	56.44

intro

related work bert gpt-3 , M-Bert, legal Bert, Greek-Bert