Argumentation Mining for Legal Texts

Nicola Amoriello - 0000952269
nicola.amoriello@studio.unibo.it

April 2022

Contents

1. Introduction

The purpose of the project was to explore a dataset created for argumentation mining tasks in a legal context and experiment on it. In fact, this topic is recently gaining a lot of interest in the community of scholars and the implementation of this in the legal context could simplify the work of professionals in the field.

My work has been to experiment using the data provided. I first took care of preprocessing the data, so as to remove present outliers and refine dirty samples. Next, I thought of various methods for how to represent the embeddings of the sentences. In the end, models were implemented to solve tasks: in particular, the task to classify sentences as premise/conclusion and the task to classify different types of arguments.

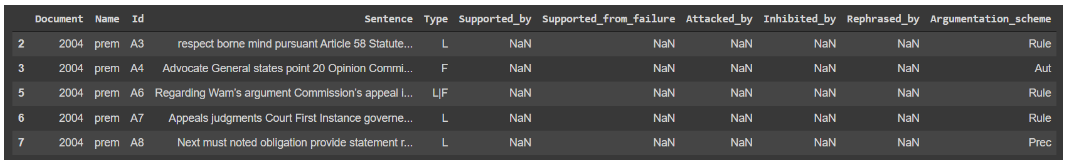
2. Data Preprocessing

2.1 Preprocessing steps

In order to build the dataset I extracted the files sent to me from the zipper archive and preprocessed it. As parts of the proprocessing pipeline, I did the following:

1. Aggregated all the information from the various json files into a single pandas DataFrame
2. I removed stopwords and punctuation from elementens in the sentence column;
3. Outlier rows (with missing or inconsistent core data) have been removed
4. Different representations of the same element have been homogenized into a single representation.

So, at the end of this preprocessing pipeline, the DataFrame looks as follows:



	Document	Name	Id	Sentence	Type	Supported_by	Supported_from_failure	Attacked_by	Inhibited_by	Rephrased_by	Argumentation_scheme
2	2004	prem	A3	respect borne mind pursuant Article 58 Statute...	L	NaN	NaN	NaN	NaN	NaN	Rule
3	2004	prem	A4	Advocate General states point 20 Opinion Commi...	F	NaN	NaN	NaN	NaN	NaN	Aut
5	2004	prem	A6	Regarding Wam's argument Commission's appeal i...	L|F	NaN	NaN	NaN	NaN	NaN	Rule
6	2004	prem	A7	Appeals judgments Court First Instance governe...	L	NaN	NaN	NaN	NaN	NaN	Rule
7	2004	prem	A8	Next must noted obligation provide statement r...	L	NaN	NaN	NaN	NaN	NaN	Prec

Figure 2.1: A view of some rows of the DataFrame.

The results of this aggregation and pre-processing work were then saved as in a pickle file so that it could be easily retrieved and used without having to repeat this process each time.

3. Background

In this chapter I will briefly list all the models which I tried to approach the Argumentation Mining tasks of the project.

3.1 Embeddings

3.1.1 TF-IDF

TF-IDF is a simple method for vectorizing sentences. To calculate the value of a word in a document, multiply the term frequency by the inverse document frequency.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

3.1.2 Sentence BERT

Sentence Bert is a state of the art framework for embedding sentences. Using it, it is possible to represent sentences in multidimensional vectors that can be compared to assess whether they have similar meaning content. It offers a large collection of pre-trained models tuned for various tasks, in fact for our purposes have been used two models already pre-trained: Sentence Bert Base that has been trained on texts of various kinds so as to be used for different purposes; Legal Sentence Bert that has been trained on legal texts and then has been studied to be used in the legal context.

3.2 Models

3.2.1 SVM

Support Vector Machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.

A support vector machine takes these data points and outputs the hyperplane that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify with a label, and anything that falls to the other the other label.

3.2.2 RNN

I tried a triple-layer Dense, with dropout layers and s always, the last layer is a Dense one, with the Softmax activation function, in order for the model to predict the class probabilities for each sample.

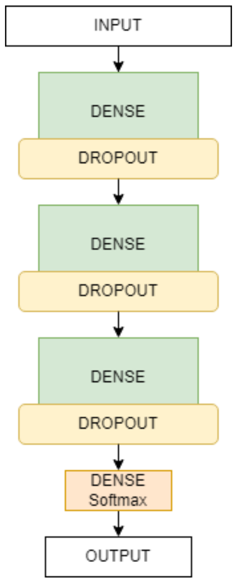


Figure 3.1: Model architecture

As suggested by the state-of-the-art, I used the Adam optimizer with its default parameters (beta1=0.9, beta2=0.999, epsilon=1e-07) to compile the model. Only the learning rate was tuned during the experiments to change the course of the training and/or to avoid being blocked a local minimum. As loss has been used the categorical crossentropy.

3.2.3 BERT

BERT is a model which makes use of transformers: in a nutshell, a transformer is a model which uses attention mechanisms to forward a more complete picture of the whole sequence from an encoder to a decoder at once rather than sequentially. BERT's architecture, then, can be described as a multi-layer bidirectional transformer encoder.

3.2.4 multilabel Model

To implement a model in order to make multilabel classification it was decided a triple-layer LSTM, with dropout layers. I prefered the LSTM layers over the standard RNN ones, due to the fact that the latter suffer from the "vanishing gradient" problem. The last layer is splitted in ten different one neuron Dense layer, with the Sigmoid activation function for each one, in order for the model to predict the class probabilities independently from each other.

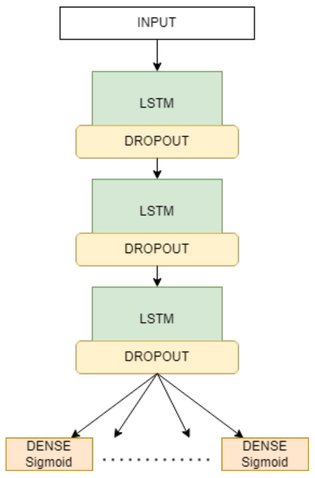


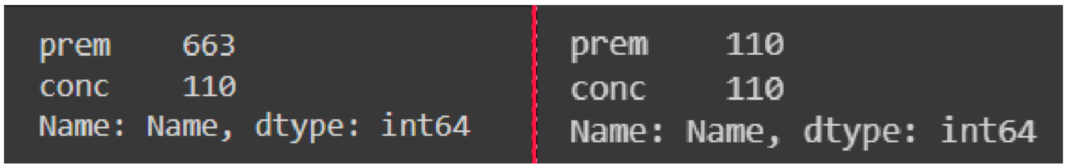
Figure 3.2: Model architecture

Even for this model, I used the Adam optimizer with its default parameters (beta1=0.9, beta2=0.999, epsilon=1e-07) to compile the model. Only the learning rate was tuned during the experiments to change the course of the training and/or to avoid being blocked a local minimum. As loss has been used the binary crossentropy.

4. Experiments

4.1 promise/conclusion classification

For the experiments it was chosen to use, in addition to the full dataset, a truncated dataset in order to have the features perfectly balanced in number of occurrences. So it was decided to use a dataset for train/val/test containing all samples that we will call "df_full" and one well balanced, but with less samples that we will call "df_balanced".



```
prem    663
conc    110
Name: Name, dtype: int64
```

```
prem    110
conc    110
Name: Name, dtype: int64
```

Figure 4.1: A view of the input data. (full dataset on the left and balanced dataset on the right)

Once the inputs were defined, various experiments were performed using the models described above and by tuning the various hyperparameters.

Below is a table of some of the experiments performed:

Df	embedding	Kernel	Gamma	C	Acc	F1
df_full	legal_bert_sentence	sigmoid	0.01	0.001	0.99	0.98
df_balanced	tfidf	linear	0.1	0.1	0.94	0.94
df_full	tfidf	linear	0.01	0.001	0.87	0.47
df_balanced	bert_sentence	linear	0.1	0.1	0.89	0.89
df_balanced	bert_sentence	poly (2)	0.1	0.1	0.48	0.33
df_balanced	bert_sentence	poly (3)	0.1	0.1	0.48	0.33
...	...	...	...	...	...	...

Table 4.1: Results of SVC implementations on the test set

Df	embedding	epochs	batch	layers	lr	l2_factor	drop	Acc	F1
df_full	bert_sentence	40	64	[128, 64, 32]	0.001	0.001	0.01	0.97	0.93
df_balanced	bert_sentence	40	64	[128, 64, 32]	0.001	0.001	0.01	0.94	0.94
df_balanced	bert_sentence	60	32	[128, 64, 32]	0.001	0.001	0.2	0.91	0.91
df_full	legal_bert_sentence	60	32	[128, 64, 32]	0.001	0.001	0.2	0.97	0.94
...	...	...	...	...	...	...	...	...	...

Table 4.2: Results of RNN implementations on the test set

Df	embedding	epochs	batch	lr	drop	Acc	F1
df_balanced	legal_bert_sentence	100	32	10e-6	0.2	0.68	0.66
df_balanced	legal_bert_sentence	100	64	10e-6	0.2	0.76	0.76
df_balanced	legal_bert_sentence	100	64	10e-6	0.2	0.76	0.76
df_full	legal_bert_sentence	100	64	10e-7	0.3	0.59	0.55
df_full	legal_bert_sentence	50	64	10e-4	0.2	0.34	0.34
...	...	...	...	...	...	...	...

Table 4.3: Results of BERT implementations on the test set

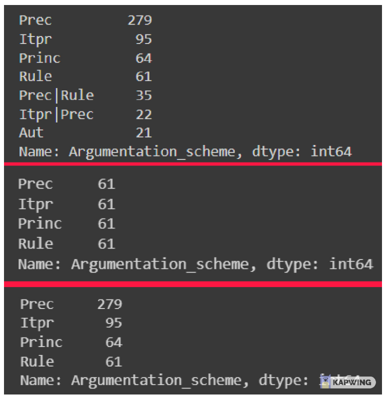
As we can see the task is performed with excellent results more or less with all embeddings and models, although it seems to perform better than the others the sigmoid SVC, probably because it is a method that performs well even with a few hundred samples

4.2 Argumentation type classification

As was done in the previous task, we decided to truncate the dataset at different points to create different distributions of samples to be used in our experiments. More specifically we decided to use, in addition to the complete dataset, a truncated dataset with a treshold of 20 instances per class, one with a treshold of 60 and one with all classes with the same number of samples. Respectively these datasets will be called hereafter "df_full", "df_thr20", "df_thr60" and "df_balanced".

Once the inputs were defined, various experiments were performed using the models described above and by tuning the various hyperparameters.

Below is a table of some of the experiments performed:



```
Prec         279
Itpr          95
Princ         64
Rule          61
Prec|Rule     35
Itpr|Prec     22
Aut           21
Name: Argumentation_scheme, dtype: int64
```

```
Prec     61
Itpr     61
Princ    61
Rule     61
Name: Argumentation_scheme, dtype: int64
```

```
Prec     279
Itpr      95
Princ     64
Rule      61
Name: Argumentation_scheme, dtype:
```

Figure 4.2: A view of the input data. (From top to bottom respectively the balanced dataset, the dataset with treshold at 20 and the dataset with treshold at 60)

Df	embedding	Kernel	Gamma	C	Acc	F1
df_thr60	legal_bert_sentence	rbf	0.01	0.001	0.73	0.60
df_thr60	tfidf	rbf	0.01	0.001	0.60	0.19
df_thr20	legal_bert_sentence	poly (3)	0.01	0.001	0.65	0.42
df_thr20	legal_bert_sentence	linear	0.01	0.001	0.64	0.55
df_thr60	bert_sentence	rbf	0.01	0.001	0.65	0.42
...	...	...	...	...	...	...

Table 4.4: Results of SVC implementations on the test set

Df	embedding	epochs	batch	layers	lr	l2_factor	drop	Acc	F1
df_thr20	tfidf	100	64	[128, 64, 32]	0.0001	0.001	0.3	0.66	0.30
df_thr20	bert_sentence	100	64	[128, 64, 32]	0.0001	0.001	0.3	0.55	0.44
df_thr60	bert_sentence	100	64	[128, 64, 32]	0.001	0.0001	0.2	0.73	0.58
df_balanced	legal_bert_sentence	100	64	[256, 128, 64]	0.001	0.0001	0.2	0.73	0.72
df_thr60	legal_bert_sentence	140	64	[256, 128, 64]	0.001	0.001	0.3	0.81	0.69
...	...	...	...	...	...	...	...	...	...

Table 4.5: Results of RNN implementations on the test set

Df	embedding	epochs	batch	lr	drop	Acc	F1
df_balanced	legal_bert_sentence	100	64	0.0001	0.3	0.57	0.57
df_thr60	legal_bert_sentence	200	64	10e-6	0.2	0.24	0.20
df_thr20	legal_bert_sentence	140	128	10e-6	0.2	0.03	0.04
...	...	...	...	...	...	...	...

Table 4.6: Results of BERT implementations on the test set

The multilabel model used for this task does not present any satisfying results. Indeed it suffers from gradiend exploding, resulting in extremely large updates to the weights of the neural network model during training, so the model is unstable and unable to learn from the training data.

5. Conclusions

As we have seen, the task of classifying sentences in argumentative texts as premises/ conclusions proved to be an easy problem to solve, as proven by the fact that all the models implemented were able to achieve from good to excellent results following the evaluation metrics. On the other hand, the task to classify the type of argument proved more difficult, never achieving good or fair results.

I think the main problem is that a multilabel classification task is difficult to solve with a dataset with few samples and among which there are no occurrences of some classes.

So future works could be to update the dataset with more samples that can adequately represent the problem domain and an approach using some state of the art models.

Bibliography

[1] Keras: the Python deep learning API. https://keras.io/

[2] Diederik P. et al. "Adam: A Method for Stochastic Optimization."(2017)

[3] Multilabel ranking metrics https://scikit-learn.org/stable/modules/model_evaluation.html#multilabel-ranking-metrics/

[4] TF-IDF https://monkeylearn.com/blog/what-is-tf-idf/

[5] SVM https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[6] Sentence Bert https://huggingface.co/sentence-transformers/stsb-bert-base

[7] Legal Sentence Bert https://huggingface.co/nlpaueb/legal-bert-base-uncased