Homework 2: Semantic Role Labeling (SRL)

Mattia Pannone
pannone.1803328@studenti.uniroma1.it

1 Introduction

This homework deals with the task of **Semantic Role Labeling (SRL)** which scope is to better understand the meaning of a sentence and the relationships between the words in it. In order to do this with this work I want to explore 4 main tasks that allow you to disambiguate a sentence by focusing on the predicates it contains and the topics related to each of them. These tasks are: **(1) Predicate Identification:** to identify within a sentence which is, or which are the predicates it contains; **(2) Predicate Disambiguation:** it is a crucial point as predicates can assume different meanings and refer to different roles based on the context of the sentence, therefore with this sub-task each predicate is associated with a specific class that can group verbs with similar syntact-semantic property. There are different resources that define predicates senses (and semantic roles for the following sub-task), the dataset provided uses VerbAtlas (Di Fabio et al., 2020). **(3) Argument identification:** having now the verbs with their meaning, it is possible to identify the arguments related to each of them; **(4) Argument Classification:** this can be more complicated because if a sentence contains more verbs, obviously there can be more arguments related to different predicates and with different meanings, therefore each argument is associated with a role (different classes defined by VerbAtlas as mentioned previously). The goal is to extract the predicate-arguments structure that captures the relationships between the words in a sentence.

An example of this four sub-tasks is: having the sentence (taken by the dev data-set): *"The Committee insists that all perpetrators of violations of the Convention be brought to justice."*, with task (1) identify two predicates: "insists", "brought" , with task (2) disambiguate the predicates in the following classes respectively: "CAUSE-MENTAL-STATE" and "BRING"; with task (3) identify the arguments of the two verbs ("Committee","that") for the first and ("perpetrators","to"); finally task (4) provide the classes of first couple ("agent","theme") and of second couple ("theme","destination") of arguments.

2 Data Representation and Organization

For these tasks, as well as for any task in any area, data is an important part, as a machine learning model learns from them and the performance it can achieve also depends on how they are organized.

In the dataset provided (for training and for testing) sentences are given with syntactic information and the relative labels with the classes of the predicates and those of the arguments, as well as their position in the sentence. To represent the different words in order to encode them in a format good for input to a neural network, I used a recent technology based on Transformers, which I will discuss later, which allows first to convert each word into an id, then the ids will be represented with a series of features by a network that allows you to contextualize the words. If for tasks (1) and (3) it was sufficient to identify each word with '1' if they were predicates or arguments and '0' otherwise, since we are dealing with identification tasks, tasks (2) and (4) are instead multi class classification task, so I created the two groups of classes between the data provided: for predicate disambiguation 306 classes were found, while for argument classification 26 classes. both groups have been added the character underscore identified by 0 to indicate that it was not a target word and the token "[PAD]" corresponding to -100 to indicate padding. About the latter, I added a length padding for each sentence in order to make all the sentences of the dataset as long as the longest sentence present.

To build the training and test set I used the "words" of the dataset, which unlike the "lemmas" take the original words present in the sentence, including cased characters; the tokenizer I used split

some compound words into root and ending, I only took into account the root. Words unknown to the tokenizer replaced with the token "[UNK]", have been left that way without deleting or replacing them.

Since one of the difficulties in the four sub-tasks was the association of different arguments to different predicates possibly present in the same sentence, for tasks (3) and (4) each sentence was duplicated as many times as there were predicates, associating each copies the unique predicate and argument labels, so as to have an input to the model consisting of a sentence and a single predicate-argument pair, (later when I talk about the model implementation I will explain how they are associated within it). Another type of sentence organization, which I used in particular for task (4), involves the input consisting of "[CLS] tokens of sentence [SEP] predicate", with "[CLS]" and "[SEP]" special tokens, the tokens of the sentence in the middle and the reference predicate at the end.

3 Model Architecture

All the models I used are based on some fundamental technologies: Transformers, the recurrent neural network and linear classification layers.

A transformer is a type of neural network architecture that was introduced in 2017 (Vaswani et al., 2017). The transformer architecture is designed to process sequential data and it uses self-attention mechanisms to weigh the importance of different parts of the input when making predictions. To date there are several types of pre-trained networks based on transformers, in particular for this homework I mainly used BERT which is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Devlin et al., 2018). In some tests I also used its derivatives RoBERTa (Liu et al., 2019) and ALBERT (Lan et al., 2019) which both have fewer parameters and therefore a lighter structure than Bert, but it has been shown in some cases they can be more performing. The kind of RNN used is BiLSTM introduced in 1997, which allowing the network to take into account both past and future context when processing sequential data (Hochreiter and Schmidhuber, 1997). The last layer for each model is a linear classifier.

In the next sections I will explore the combination of these stages used in the four tasks mentioned above.

3.1 Predicate Identification

This simple identification task uses only two layers: a transformer layer with BERT, from which I take as output the average of the last 4 layers of the representation it provides, and then pass it to a single linear layer (tests were also made with two linear layers) which returns numbers for each token which will be approximated by '1' if it is a predicate, '0' otherwise.

3.2 Predicate Disambiguation

Starting from the previous model two elements in particular have been added between the transformer and the linear classifier: the first is a representation of where the predicates were, therefore a tensor has been added which for each token of the input concatenated a '1' if there was a predicate at that position, '0' otherwise. The other element added was 2 layers of BiLSTM. This time the classifier will return a set of probabilities for each token, so the argmax will assign a class to that token.

3.3 Argument Identification

This model is almost equal to the two previous ones, in fact the architecture I used is equal to that for predicate disambiguation, but the linear classifier return only one element such as in predicate identification.

3.4 Argument Classification

Also this model is almost equal to the previous one and so to the first two but in this case the new features i concatenate to specify the position of predicates in previous tasks, this time is to specify the positions of arguments (maintaining the '1' if in that position there was an argument, '0' otherwise). In this model I also concatenate a new feature:

$\forall w \in Sentence \quad Mean(Features(w)) - Mean(Features(Predicate))$

The classifier will return a set of probabilities for each token, so the argmax will assign a class to that token.

4 Training and Evaluation

Evaluation of different models are made with the F1 score that is the harmonic mean of precision and recall, this metric can be better with respect to accuracy because an high accuracy do not mean necessarily good performance in that, as shown in Figure 1 and Figure 2 there an imbalanced distribution of data with respect to class, in particular

there is the class '_' which denote in all task both that the correspond token is not a predicate and not an argument, also there an high number of tokens ad padding in many sentence, this will be not considered in the classifications tasks thanks to the parameter ignore index of the loss function. All models are trained with a batch size of the dataset of 32. For the identification tasks ((1) and (3)) I used the 'Sigmoid' activation function and the Mean Squared Error loss, while for classification tasks ((2) and (4)) I used 'Softmax' in the test phase (no one neede when pass data to the loss fucntion) and the Cross Entropy loss function. All tasks was trained mainly with 10 epochs (with early stopping active during training) and different tries with different results are given by the decision to fine-tune the transformer model during the train or freeze it: in some cases the fine-tune improved the result (for example task (1) reach 92% in 10 epochs without Bert fine-tune, while it reach the same performance in only 2 epochs with the fine-tune). In other cases the Bert fine-tune in my models stopped the learning of the models with the stuck of the loss and the performances down until 0. Since in all models I used the Adam optimizer, the last problem described was partially solved by changing the learning rate and the weight decay (a method for normalization of data), both set in some cases between 0.001 and 0.00001.

In model of task (1) the input for the models is a dictionary with the list of tokens id for each word and an attention mask; in model of task (2) and (3) the same input but adding the list of predicates; moreover for task (3) i trained the models in two ways: first of all duplicating the sentences as many times as the number of predicates, so that for each sentence we have only a predicate and we have to predict only the arguments for that predicate, For task (3) I also trained a model in a mixed way: 5 epochs with non-duplicate data and another 5 first, then 15, with duplicate data, this allowed to increase the performance by +1% (88% - ¿ 89%). For model (4) I also added another way for training: 10 epochs with normal data duplicated and 10 epochs with data duplicated and formatted such that: "[CLS] tokens_id sentence [SEP] predicate"[*]. With this mixed method I not improved (only equalized) the performance with respect the first training, but the train with data formatted only as [*], the performance improved of +2% (84% -¿ 86%).

All the tries are done performing each task in an independent way between them, with the correct labels from the dataset, so the performance are higher with respect to execute the tasks in order as pipeline (1)+(2))+(3)+(4). As said I reach 92% in Predicate identification, while I reach 81% in predicate disambiguation, 89% in Argument Identification and 84% in Argument Classification. The figures [3,5,6,8] show the training of these models while the tables [1,2,3,4] show some comparisons.

Figures [4,7] show the confusion matrix of the models with which a reached higher performance with each task, in particular the confusion matrix for tasks (2) and (4)are not present because very hard to read with more than 300 classes for the first and 27 for the second to represent, while those for tasks (1) and (2) are much more readable. In all confusion matrices the rows represent the number of true label for each class while each column represent the number of predictions for each class, so in the diagonal there are the so called true positive. For tasks (1) and (3) we can see that model reach good performances as it the number of true positive predicted is high, shown also by the intensity of the blue color in the matrix.

5 Not only BERT and not only English

As mentioned in a previous section, for each task I retrained the model of each one who achieved the best performances but substituting BERT, first ALBERT and then RoBERTa. Having left all the hyperparameters fixed, performance has not improved for any model, but probably playing more with different configurations would have room for improvement. (Table [5]).

Another test on the models was to do transfer learning on the two other natural languages, Spanish and French, although the performance results are rather low, it must be said that the training was carried out only with the BERT version for the 'English and pulling the previous models with 10 other epochs but with the data of the new languages (organized and represented in the same way). (Table [6]).

6 Conclusions

I studied on these four task to try to reach good performances, however my models can be improved by revisit the model architectures, dealing with some hyperparameters and dealing with organization and representation of data.

References

- Di Fabio, A., Conia, S., & Navigli, R. (2020). VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling. Sapienza University of Rome, Italy.
- Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.
- Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692.
- Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2019). ALBERT: A lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

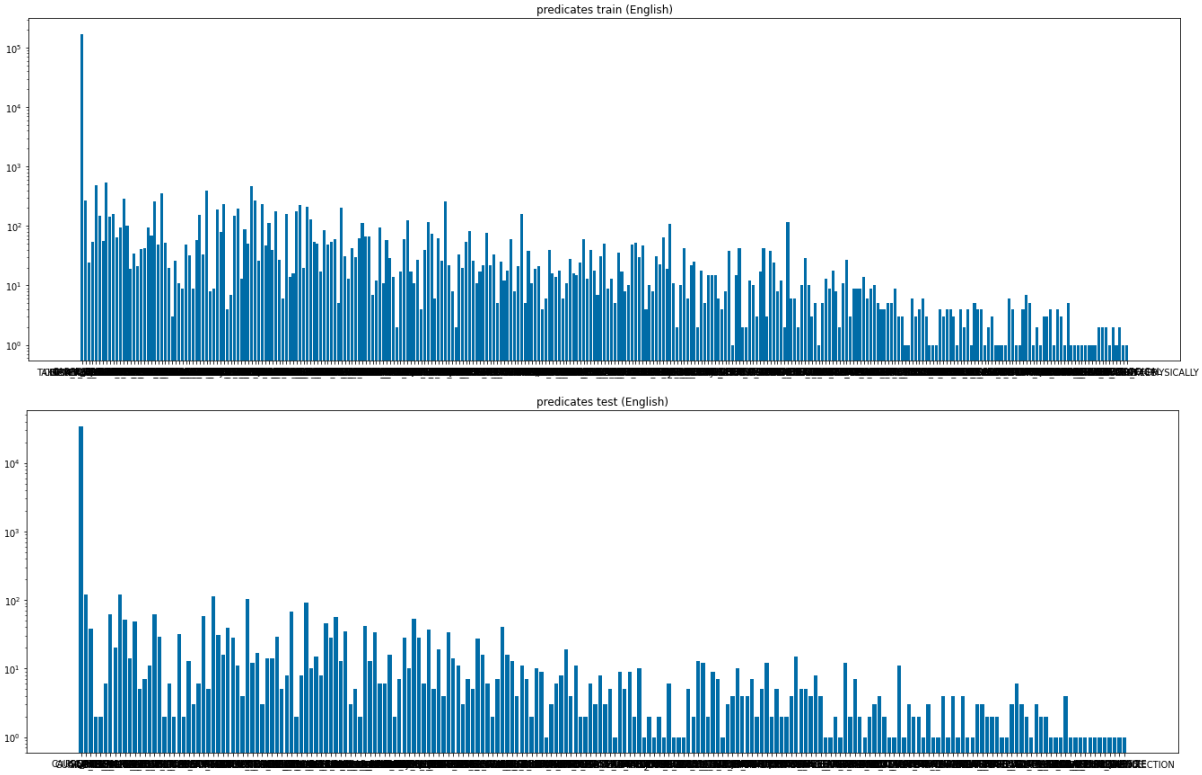


Figure 1: The two above images show the number of classes for predicate disambiguation in the train set and in the test set. The first long bar represent the "_" class which indicates that it is not a predicate. It can be seen that some classes are more present than others, which can affect training and testing.

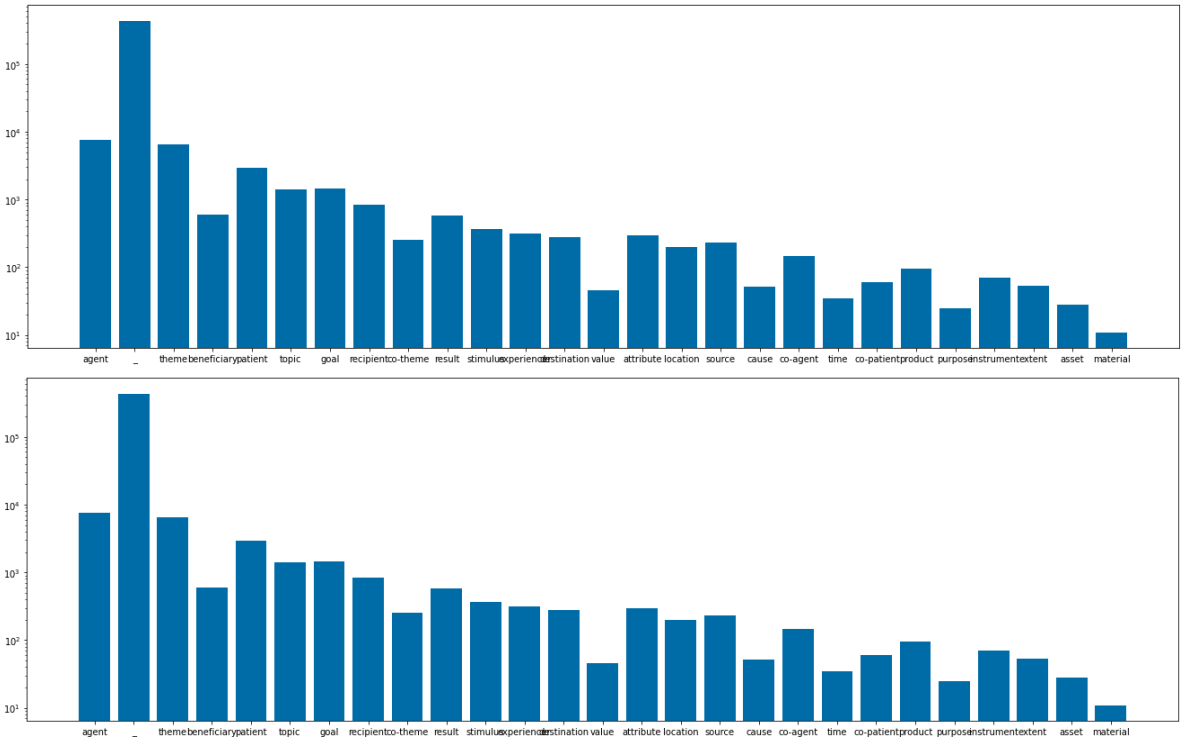
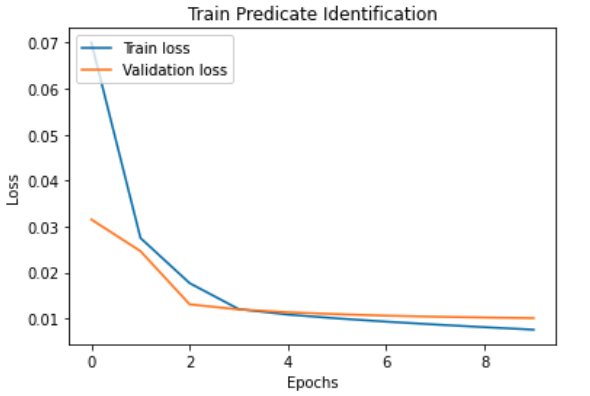
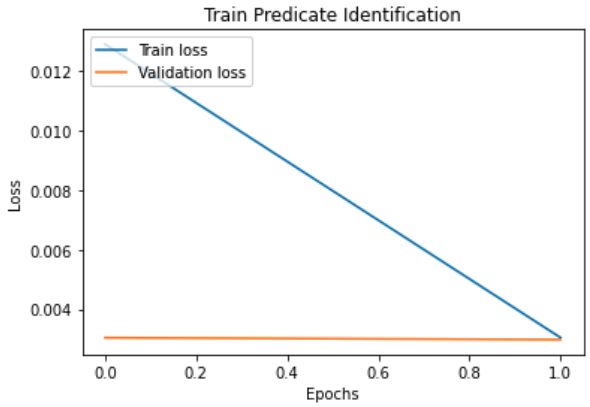


Figure 2: The two above images show the number of classes for argument classification in the train set and in the test set. The first long bar represent the "_" class which indicates that it is not a predicate. It can be seen that some classes are more present than others, which can affect training and testing.



(a)



(b)

Figure 3: (a) represent the train in 10 epochs of model for predicate identification without the fine-tuning of Bert, while (b) represent the training with fine tune of Bert in 2 epochs.

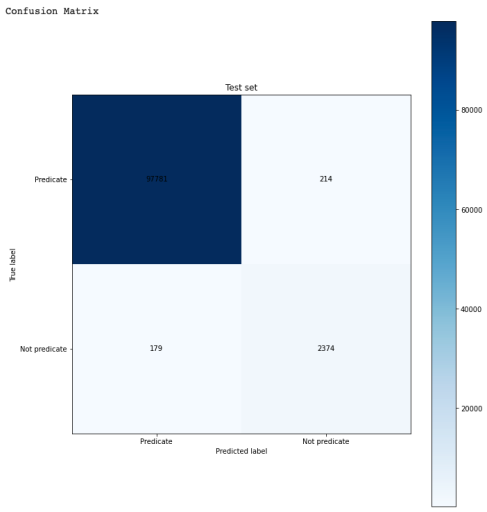
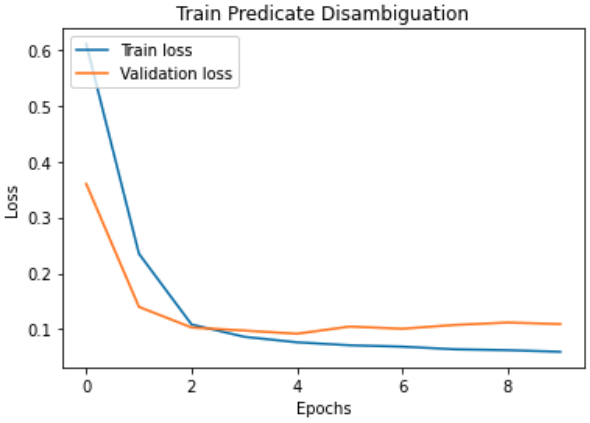
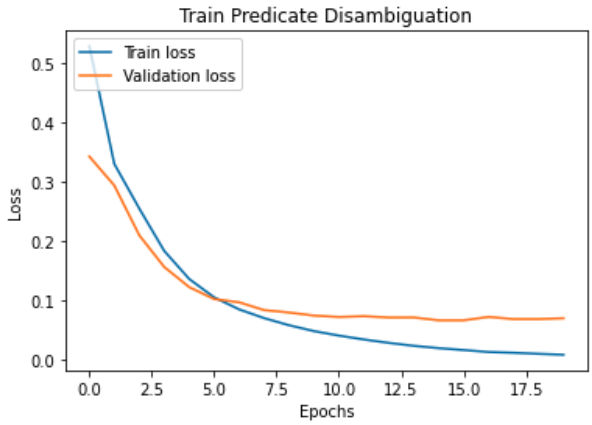


Figure 4: This is the confusion matrix for predicate identification task. It can be seen that there are many true positives from the intensity of the blue color in first row first column.

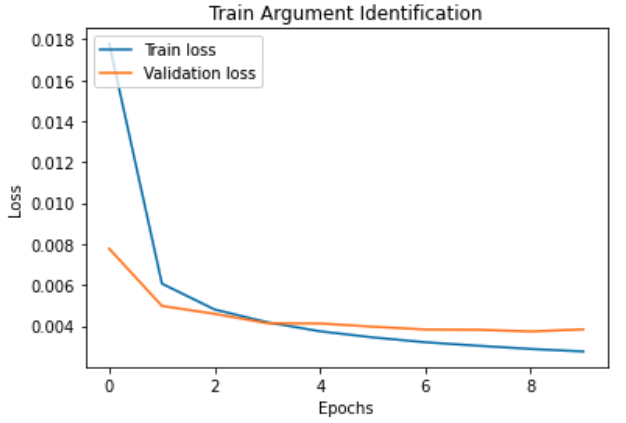


(a)

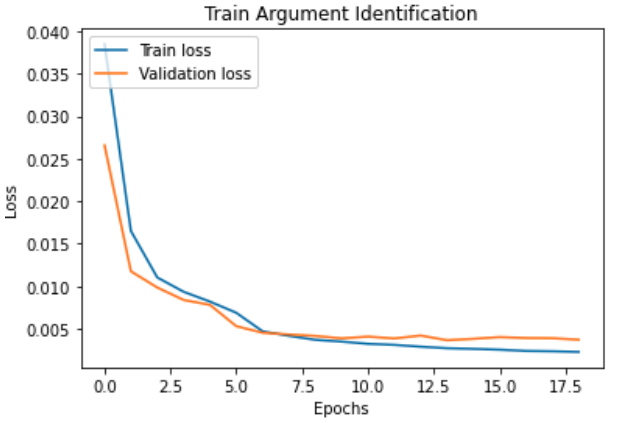


(b)

Figure 5: (a) Train of predicate disambiguation model with a layer of Bert (fine-tuned) and one linear classifier in 10 epochs while (b) Train of predicate disambiguation model with Bert+BiLSTM+linear classifier with tagged predicates and Bert not fine-tuned, training stopped with early stop at 20 epochs.



(a)



(b)

Figure 6: (a) Train of argument identification model with a layer of Bert (not fine-tuned) + BiLSTM+linear classifier with tagged predicates in 10 epochs. (b) Train of argument identification model with a layer of Bert (not fine-tuned) BiLSTM + linear classifier with tagged predicates in 5 epochs with not duplicated data and 15 epochs with duplicated data.

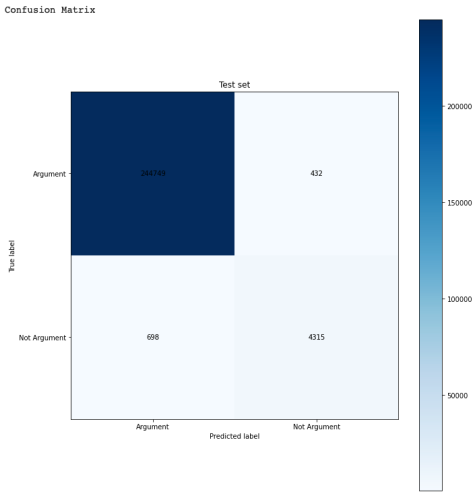
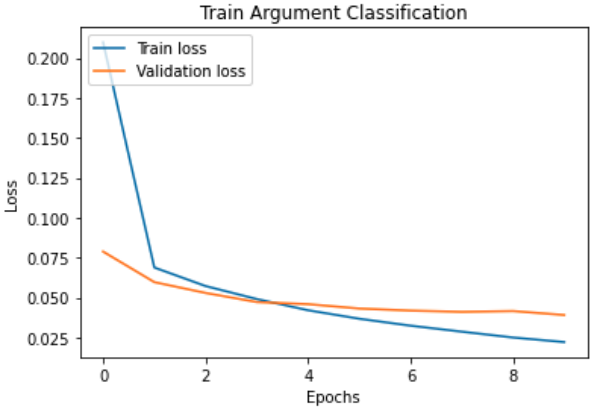
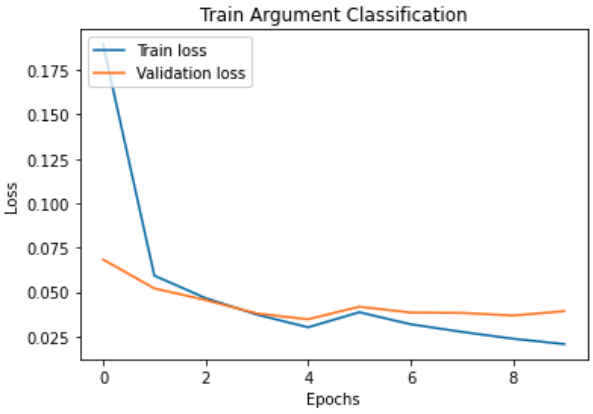


Figure 7: This is the confusion matrix for argument identification task. It can be seen that there are many true positives from the intensity of the blue color in first row first column.



(a)



(b)

Figure 8: (a) Train of argument classification model with a layer of Bert (not fine-tuned) + BiLSTM + linear classifier with tagged predicates and difference between mean of words and predicate, trained with normal duplicated data (b) Train of argument classification model with a layer of Bert (not fine-tuned) + BiLSTM + linear classifier with tagged predicates and difference between mean of words and predicate, trained with normal duplicated data first and than with duplicated data formatted so: [CLS] sentence [SEP] predicate

Model	Add features	Bert fine-tune	epochs	F1
Bert + 2 linear	-	no	10	**0.92**
Bert + 1 linear	-	no	10	**0.84**
Bert + 1 linear	-	yes	2	**0.92**

Table 1: F1 scores of 3 different trained models. Here you can see several comparisons: in the first 2 models with the same epochs, an extra linear layer makes the difference; while between the third and the first 2, Bert's finetuning makes the difference.

Model	Add features	Bert fine-tune	epochs	F1
Bert + 1 linear	-	yes	10	**0.78**
Bert + 2 linear	Tag predicates with '1'	no	10	**0.78**
Bert + 2 BiLSTM + 1 linear	Tag predicates with '1'	no	10	**0.79**
Bert + 2 BiLSTM + 1 linear	Tag predicates with '1'	no	20	**0.81**

Table 2: F1 scores of 3 different trained models. We can see that there is a small improvement in inserting 2 layers of BiLSTM and adding the tag for the predicates, an improvement that grows a little more by doubling the training epochs.

Model	Add features	Bert fine-tune	epochs	F1
Bert + 2 BiLSTM + 2 linear	-	no	10	**0.44**
Bert + 2 BiLSTM + 1 linear	Tag predicates with '1'	no	10	**0.88**
Bert + 2 BiLSTM + 1 linear	Tag predicates with '1'	no	5 with data not duplicated 15 with data duplicated	**0.89**

Table 3: F1 scores of 3 different trained models. We can see how the performance improve a lot first adding the new feature with the tag of predicates, then there is a small improvement doing a mixed train with data not duplicated and data duplicated.

Model	Add features	Bert fine-tune	epochs	F1
Bert + 2 BiLSTM + 1 linear	Tag roles with '1'	yes	20	**0.73**
Bert + 2 BiLSTM + 1 linear	Tag roles with '1' and mean(word)-mean(predicate)	no	20	**0.84**
Bert + 2 BiLSTM + 1 linear	Tag roles with '1' and mean(word)-mean(predicate)	no	10 with normal duplicated data 10 with duplicated data as [CLS] tokens [SEP] pred	**0.84**

Table 4: F1 scores of 3 different trained models. We can see that by freezing the fine tune of Bert this time the performance improve, (also by adding a new feature given by the difference between a word and the predicate, in addition to the role tag). There are no improvements doing a mixed training with normal duplicated data and duplicated data formatted as: [CLS] token sentence [SEP] predicate.

	Transformer finetune + BiLSTM	Transformer fine-tune and no BiLSTM	No Transformer fine-tune + BiLSTM	No Transformer fine-tune and no BiLSTM
BERT	0.49	0.77	0.79	0.78
ALBERT	0.79	0.78	0.67	0.64
RoBERTa	0.23	0.71	0.75	0.76

Table 5: F1 scores of predicate disambiguation task trained for 10 epochs with three different pre-trained transformers based on BERT. Since all the towed models have the same configuration apart from the fine-tune of the different models and the presence of the BiLSTM layer, it can be seen that each model needs a different configuration to add high performances.

	Predicate Identification	Predicate Disambiguation	Argument Identification	Argument Classification
English	0.92	0.81	0.89	0.84
Spanish	0.62	0.27	0.68	0.62
French	0.59	0.27	0.68	0.57

Table 6: F1 scores of all task for English, Spanish and French. English is the native language with which the models were trained, while for Spanish and French a simple transfer learning was used where the initial models (those that achieved the best performances) were retrained but on datasets with the respective languages for 20 epochs. Obviously the scores are lower than in the mother tongue, also due to the fact that no particular techniques have been used, however it can be seen that the performances in the identification tasks which are simpler than in the classifications, the scores are higher.