Semantic Role Labeling

Omar Bayoumi
Sapienza University of Rome
Natural Language Processing
Homework 2
bayoumi.1747042@studenti.uniroma1.it

1 Introduction

In NLP, **Semantic Role Labeling (SRL)** is the ability to assign the semantic role that a single word plays within the sentence, such as agent, goal, or result, with the purpose of understanding the meaning of the sentence. Thus, the main goal is to associate word arguments for each predicate within a sentence. So the target tasks are: *Predicate Identification; Predicate Disambiguation; Argument Identification; Argument Classification*

2 Preprocess

Reading an entry of the dataset we realize that there is a lot of information such as: *predicates, lemmas, words* and so on. We need a **preprocess**.

2.1 Base

First, sentences are duplicated by creating copies as many as there are predicates, so that each copy specializes on the single predicate. Then each sentence is tokenized using the tokenizer offered by **HuggingFace** [0]. This tokenization assigns ids to words and, if necessary, divides uncommon words using as a prefix the part of the word that is very frequent in the corpus. Passing all sentences also adds padding at the end of the sentence to keep the same size as the longest sentence. In addition to this, an index array is created to **identify the predicate position**: if the word is a padding the index is 0; if its not a predicate the index is 1; if its a predicate the index is 2. The special sentence start/end character, padding, and the part after the first in the splitted word are **considered as padding** and therefore will not affect in the loss function.

2.2 Add the part of speech tagging (POS)

Taking all **pos_tags** into consideration, an encoding was added to the input that tells for each word which pos_tag it has. For "padding" words, a special *PAD* tag was used so that it does not affect the loss function

2.3 Base exploiting part of speech tagging and VerbAtlas (Extra)

As an extra I integrated **VerbAtlas** within my preprocess improving the results. **VerbAtlas** is a hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from **BabelNet** into semantically-coherent frames [1].

2.3.1 Add the VerbAtlas

Following the idea of *Jangseong bae and changki lee* [2], at the end of each sentence I added **"[SEP]", the predicate lemma and the first two arguments of that type of predicate found in VerbAtlas**. The reason why this method greatly improved the results [*Figure 1*] is mainly because at the end of the sentence there are the lemma of the predicate we are considering and more importantly the two probable roles that we will have to predict

3 Model

The models used are all based on a **transformer**. A transformer adopts the mechanism of **self-attention**, weighting the meaning of each part of the input. Thus, unlike a common RNN, the input is not passed one piece at a time but all at once, parallelizing and also greatly reducing the training time.

3.1 Base model

My basic model [*Figure 6*] consists of **an Eembedding layer, a Transformer, a BLSTM and a Classifier**. The Embedding layer aims to encode whether a given word **is a predicate or not** by using as input the index array described in the base preprocess [*2.1 Base*]. The Transformer encodes the sentence. The output of the **Transformer and**

Embedding are concatenated and passed to the BLSTM to perform further encoding of this sequence. The output of the BLSTM is passed to the classifier, which is responsible for predicting word roles for that predicate within the sentence.

3.2 Add the POS embedding

To the previous model, an additional Embedding layer has been added [*Figure 7*]. This Embedding is intended to **encode the pos_tag** of each word within the sentence; the encoding at position 0 is the padding and will be ignored in the loss calculation. The idea behind it is to give additional information about the sentence to allow the model to understand the connections present in the sentence by the different pos_tags. The output of this Embedding layer is **concatenated together with the output of the Transformer and Embedding that identifies the predicate**, and then everything is passed to the BLSTM and then classifier. This model was also used with the augmented input of VerbAtlas since the input is changed and not the model.

4 Experiments

The experiments were performed on **Google Colab** using the hyperparameters described in the *Table 1* (step 3-4) for a specific model; looking for the best ones through a random search. The best results are reported in the various notebooks. One thing to mention is that the optimizer used was **Adam**, giving better results, most likely because Adam maintains a per-parameter learning rate that improves performances on problems with sparse gradients. Another hyperparameter I want to mention is the choice of **Transformer type**: tests were conducted using a **case-sensitive** and an **uncase-sensitive** version. The best results were obtained with the **uncase version**. Training was carried out for up to 100 epochs using the **early stopping technique**, and as a result training an average of 30 epochs. I also used the **dropout** with a value of, 0.5 for the BLSTM and 0.2 for the rest of the net, to try to **reduce the overfitting**. As loss function I used the **Cross Entropy Loss**. The plots *Figure 8 Figure 9 Figure 10* show the **loss function** of the best checkpoints of the three types of models/precesses used. As can be seen, the use of VerbAtlas helped a lot in bringing down the loss function.

4.1 Results

All results on the tests are shown in the *evaluation.ipynb* notebook. In *Figure 1* are shown the results obtained from different models using different hyperparameters [*Table 1*] as well. As the loss function advised the model with **best results is the one with VerbAtlas** with a +2% for *Argument Identification*, but especially a +7% in *Argument Classification* w.r.t. the second best model. Most likely the addition at the end of the arguments recommended by VerbAtlas helped a lot to disambiguate them. Another thing to note is how the addition of POS embedding helped increase the results, even though slightly, by about 2%. Plotted in *Figure 11 Figure 12* are the **confusion matrices** for the two tasks for the best model. One thing to note in the confusion matrix for *Argument Classification* is that roles such as: instrument, asset, material, time, purpose and so on, have bad results because there are few, or no, samples of that type in the train dataset. Instead, as can be seen in the confusion matrix for *Argument Identification*, the model was able to identify the arguments very well. Confusion matrix for *Argument Identification* and *Argument Classification* for models **not** using VerbAtlas are shown in *Figure 13 Figure 14* (base model plus POS Embedding) *Figure 15 Figure 16* (base model). As can be seen for *Argument Identification*, the arguments found are fewer than those found using VerbAtlas. Because of this, roles such as **product and cause**, with the use of VerbAtlas, go from being classified correctly almost never to being classified correctly almost always.

5 Extras

As an extra, I transferred the knowledge learned from English into the other languages and implemented all four steps of the algorithm.

5.1 Training in other languages

First, I tried **three types** of approaches with the **Spanish dataset**; the results are shown in the *Figure 2*. The first approach was to use a model exactly identical to the one used for the English task, however, I got the worst results. This was most likely because the transformer used was specialized for the English language. Therefore, I replaced the "English" transformer with a **Multilingual** one. The results improved by about 8%. The best compromise in the end was to do **fine tuning** on the best English checkpoint, which even though it uses

an "English" transformer, is able to give the best results. This is certainly because it will split the Spanish words in such a way that the transformer will recognize them and use this information to give fairly acceptable predictions. Once I realized that the best technique is to do fine tuning from the best checkpoint, I applied this technique to **all three types of models/preprocesses described earlier for both Spanish and French**. All results in all languages for *Argument Identification* and *Argument Classification* are shown in *Figure 3*.

5.2 The whole algorithm

The whole algorithm is based on **4 steps**: the first step deals with **finding the predicates** within the sentences; after that, knowing the position of the predicates, it is necessary to **disambiguate them** by making explicit what type of predicate it is; having obtained predicate position and type, we can move on to the third step in which **for each predicate** we have to **find its arguments**; and finally once the arguments have been found, we have to **classify them**. These steps were implemented considering the classifications made by VerbAtlas [1]. The purpose of the project was to carry out the sequences of **steps 1-2-3-4, 2-3-4 and 3-4**. The last one has already been described and the model used is one that exploits VerbAtlas. For the other two types of sequences it was necessary to make two models, one for step 1 and one for step 2. Since these sequences lack knowledge about predicates, it was necessary to remove from the model that realizes steps 3-4 the knowledge about pos_tagging since at the predicates, for about 98% of the cases, there is the pos_tag "VERB" which would be like using a ground truth.

5.2.1 Model that implement step 1

The model is very similar to the others mentioned above [*Figure 5*]. A Transformer takes the tokenized sentence as input, without the addition of any extra structure, encoding the sentence. The output is passed to a BLSTM that encodes the sequence and then obtains a prediction through a Classifier layer. As can be seen from *Figure 4*, the results were **excellent in all languages** for the *Predicate Identification* task.

5.2.2 Model that implement step 2

The model for step two is the same as the base model [*Figure 6*] [3.1 Base model] since **from step 1** the position of predicates is available. As can be seen from the results in *Figure 4* the results were very good in English but not very good in the other languages. However, the results obtained in the other languages were even worse but were improved by **fine tuning from the English checkpoint**.

5.2.3 Results for the sequence 1-2-3-4, 2-3-4 and 3-4

Figure 4 shows the results of all sequences: **3-4, 2-3-4, 1-2-3-4**. It can be seen how the influence of having predicted rather than "gold" information affects the final result. However, considering the final result I cannot say that I am not satisfied, especially for the English results to prove the fact that such an algorithm can generate excellent results for this kind of problem.

6 Conclusion

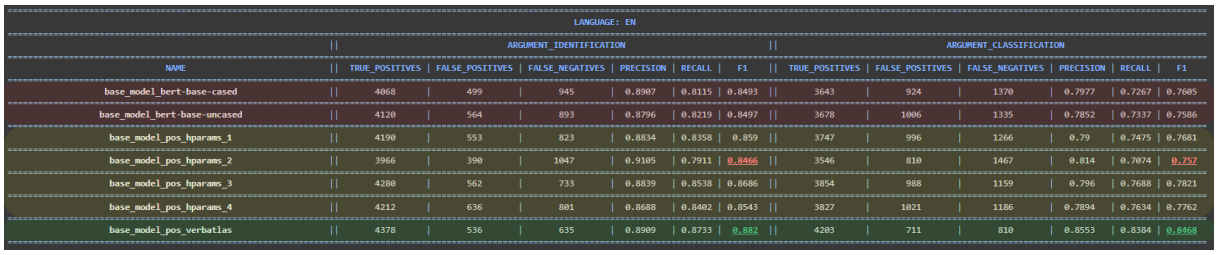
In conclusion, from this project I learned how powerful the transformers tool is in terms of both efficiency and results, and how in general we train a network to analyse a sequence of data by **decomposing the problem into multiple steps**. By unfolding steps 1 and 2 of the algorithm, I also noticed that perhaps, making a model that realizes step 3, and later using the output of step 3 in a model that realizes only step 4, would bring better results. Considering step 1 that without any information can recognize predicates very well, there is nothing to prevent me from thinking that if I had made a network that only recognized arguments I would have gotten equally good results. Reading some papers I noticed that networks well or poorly have these structures but use very high hidden spaces with a much larger dataset. Most likely with more computing power and a dataset that explores more samples, simply increasing the number of neurons could have generated much better results.

References

[0]. Hugging face community.

[1]. Verbatlas.

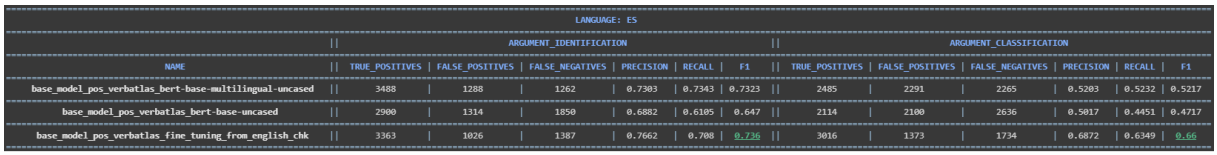
[2]. Jangseong bae and changki lee. 2022. korean semantic role labeling with bidirectional encoder representations from transformers and simple semantic information. *Applied Sciences*, 12(12).



LANGUAGE: EN

	ARGUMENT_IDENTIFICATION					ARGUMENT_CLASSIFICATION				
NAME	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1
base_model_bert-base-cased	4068	499	945	0.8907	0.8115	0.8493	3643	924	1370	0.7977	0.7267	0.7605
base_model_bert-base-uncased	4120	564	893	0.8796	0.8219	0.8497	3678	1006	1335	0.7852	0.7337	0.7586
base_model_pos_hparams_1	4190	553	823	0.8834	0.8358	0.859	3747	996	1266	0.79	0.7475	0.7681
base_model_pos_hparams_2	3966	390	1047	0.9105	0.7911	0.8466	3546	810	1467	0.814	0.7074	0.757
base_model_pos_hparams_3	4280	562	733	0.8839	0.8538	0.8686	3854	988	1159	0.796	0.7688	0.7821
base_model_pos_hparams_4	4212	636	801	0.8688	0.8402	0.8543	3827	1021	1186	0.7894	0.7634	0.7762
base_model_pos_verbatlas	4378	536	635	0.8909	0.8733	0.882	4203	711	810	0.8553	0.8384	0.8468

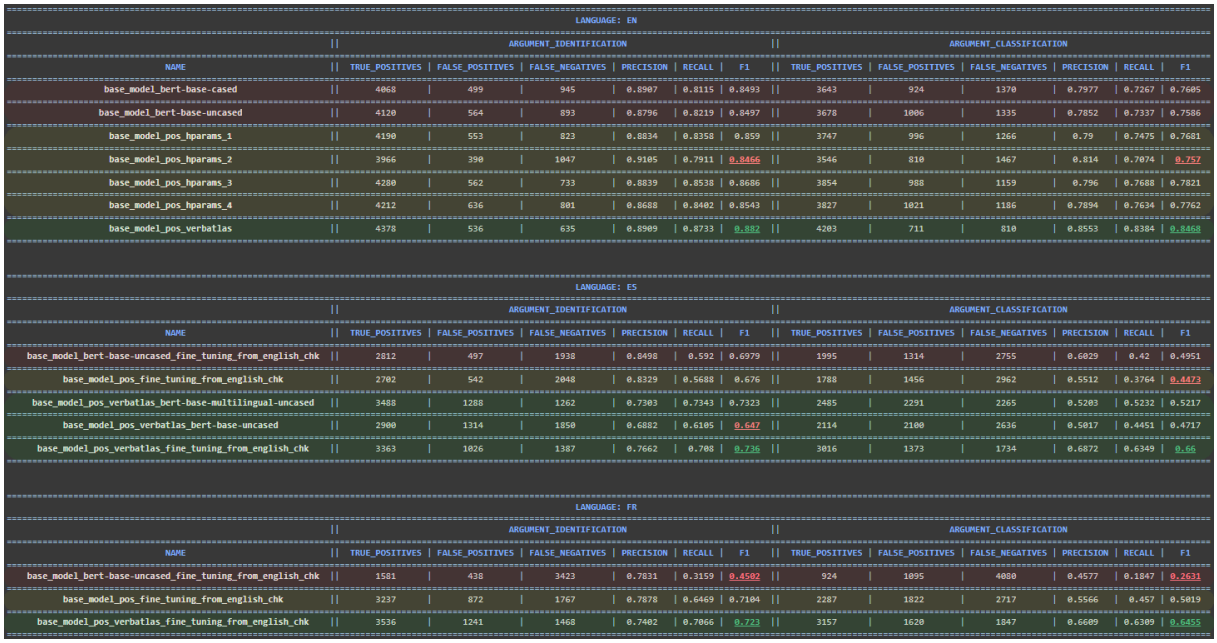
Figure 1: Results of the three types of models/preprocesses. Highlighted in red is the **base model**, in yellow is the **base model + using pos**, in green is the **base model + using pos + VerbAtlas**. All hyperparameters are described in the Table 1.



LANGUAGE: ES

	ARGUMENT_IDENTIFICATION					ARGUMENT_CLASSIFICATION				
NAME	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1
base_model_pos_verbatlas_bert-base-multilingual-uncased	3488	1288	1262	0.7303	0.7343	0.7323	2485	2291	2265	0.5203	0.5232	0.5217
base_model_pos_verbatlas_bert-base-uncased	2900	1314	1850	0.6882	0.6105	0.647	2114	2100	2636	0.5017	0.4451	0.4717
base_model_pos_verbatlas_fine_tuning_from_english_chk	3363	1026	1387	0.7662	0.708	0.736	3016	1373	1734	0.6872	0.6349	0.66

Figure 2: Results in Spanish with the three different types of approaches.



LANGUAGE: EN

	ARGUMENT_IDENTIFICATION					ARGUMENT_CLASSIFICATION				
NAME	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1
base_model_bert-base-cased	4068	499	945	0.8907	0.8115	0.8493	3643	924	1370	0.7977	0.7267	0.7605
base_model_bert-base-uncased	4120	564	893	0.8796	0.8219	0.8497	3678	1006	1335	0.7852	0.7337	0.7586
base_model_pos_hparams_1	4190	553	823	0.8834	0.8358	0.859	3747	996	1266	0.79	0.7475	0.7681
base_model_pos_hparams_2	3966	390	1047	0.9105	0.7911	0.8466	3546	810	1467	0.814	0.7074	0.757
base_model_pos_hparams_3	4280	562	733	0.8839	0.8538	0.8686	3854	988	1159	0.796	0.7688	0.7821
base_model_pos_hparams_4	4212	636	801	0.8688	0.8402	0.8543	3827	1021	1186	0.7894	0.7634	0.7762
base_model_pos_verbatlas	4378	536	635	0.8909	0.8733	0.882	4203	711	810	0.8553	0.8384	0.8468

LANGUAGE: ES

	ARGUMENT_IDENTIFICATION					ARGUMENT_CLASSIFICATION				
NAME	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1
base_model_bert-base-uncased_fine_tuning_from_english_chk	2812	497	1938	0.8498	0.592	0.6979	1995	1314	2755	0.6029	0.42	0.4951
base_model_pos_fine_tuning_from_english_chk	2702	542	2048	0.8329	0.5688	0.676	1788	1456	2962	0.5512	0.3764	0.4473
base_model_pos_verbatlas_bert-base-multilingual-uncased	3488	1288	1262	0.7303	0.7343	0.7323	2485	2291	2265	0.5203	0.5232	0.5217
base_model_pos_verbatlas_bert-base-uncased	2900	1314	1850	0.6882	0.6105	0.647	2114	2100	2636	0.5017	0.4451	0.4717
base_model_pos_verbatlas_fine_tuning_from_english_chk	3363	1026	1387	0.7662	0.708	0.736	3016	1373	1734	0.6872	0.6349	0.66

LANGUAGE: FR

	ARGUMENT_IDENTIFICATION					ARGUMENT_CLASSIFICATION				
NAME	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1	TRUE_POSITIVES	FALSE_POSITIVES	FALSE_NEGATIVES	PRECISION	RECALL	F1
base_model_bert-base-uncased_fine_tuning_from_english_chk	1581	438	3423	0.7831	0.3159	0.4502	924	1095	4080	0.4577	0.1847	0.2633
base_model_pos_fine_tuning_from_english_chk	3237	872	1767	0.7878	0.6469	0.7104	2287	1822	2717	0.5566	0.457	0.5019
base_model_pos_verbatlas_fine_tuning_from_english_chk	3536	1241	1468	0.7402	0.7066	0.723	3157	1620	1847	0.6609	0.6309	0.6455

Figure 3: Results of the three types of models/preprocesses in all language. Highlighted in red is the **base model**, in yellow is the **base model + using pos**, in green is the **base model + using pos + VerbAtlas**. All hyperparameters are described in the Table 1.

Language	Model name	lr	Transformer lr	Batch	Hidden dim BLSTM	Predicate Embedding dim	POS Embedding dim	Transformer name
				Step 1				
All	base model without predicate embedding	4e-4	4e-5	32	200	-	-	bert-base-uncased
				Step 2				
English	base model	1e-3	1e-4	32	200	200	-	bert-base-uncased
Spanish/French	fine tune from english checkpoint	1e-3	1e-4	32	200	200	-	bert-base-uncased
				Step 3-4				
English	base model hparams_1	1e-3	1e-4	80	200	128	-	bert-base-cased
English	base model hparams_2	1e-3	1e-4	80	200	128	-	bert-base-uncased
English	base model + POS hparams_1	1e-3	1e-4	32	200	128	128	bert-base-uncased
English	base model + POS hparams_2	1e-3	1e-4	80	200	128	128	bert-base-uncased
English	base model + POS hparams_3	1e-3	1e-4	32	200	200	200	bert-base-uncased
English	base model + POS hparams_4	8e-4	1e-4	32	1600	200	200	bert-base-uncased
English	base model + POS + VerbAtlas	1e-3	1e-4	32	200	128	128	bert-base-uncased
Spanish/French	fine tune from base model	5e-4	5e-5	32	200	128	-	bert-base-uncased
Spanish/French	fine tune from base model + POS hparams_3	5e-4	5e-5	32	200	200	200	bert-base-uncased
Spanish	base model + POS + VerbAtlas hparams_1	1e-3	1e-4	32	200	128	128	bert-base-uncased
Spanish	base model + POS + VerbAtlas hparams_2	1e-3	1e-4	32	200	128	128	bert-base-multilingual-uncased
Spanish/French	fine tune from base model + POS + VerbAtlas	5e-4	5e-5	32	200	128	128	bert-base-uncased

Table 1: All hparams for all models/preprocesses for all algorithm steps

LANGUAGE: EN

NAME	F1 SCORE PREDICATE_IDENTIFICATION	F1 SCORE PREDICATE_DISAMBIGUATION	F1 SCORE ARGUMENT_IDENTIFICATION	F1 SCORE ARGUMENT_CLASSIFICATION
Result_1234	0.9482	0.7943	0.8481	0.7808
Result_234	-	0.8309	0.8779	0.8072
Result_34	-	-	0.882	0.8468

LANGUAGE: ES

NAME	F1 SCORE PREDICATE_IDENTIFICATION	F1 SCORE PREDICATE_DISAMBIGUATION	F1 SCORE ARGUMENT_IDENTIFICATION	F1 SCORE ARGUMENT_CLASSIFICATION
Result_1234	0.9176	0.4465	0.6773	0.5078
Result_234	-	0.4661	0.7256	0.5422
Result_34	-	-	0.736	0.66

LANGUAGE: FR

NAME	F1 SCORE PREDICATE_IDENTIFICATION	F1 SCORE PREDICATE_DISAMBIGUATION	F1 SCORE ARGUMENT_IDENTIFICATION	F1 SCORE ARGUMENT_CLASSIFICATION
Result_1234	0.8821	0.4179	0.6586	0.4889
Result_234	-	0.4357	0.7099	0.5177
Result_34	-	-	0.723	0.6455

Figure 4: The results obtained in all languages for *Predicate Identification*, *Predicate Disambiguation*, *Argument Identification* and *Argument Classification* taking into consideration the sequences of **step 1-2-3-4, 2-3-4 and 3-4**.

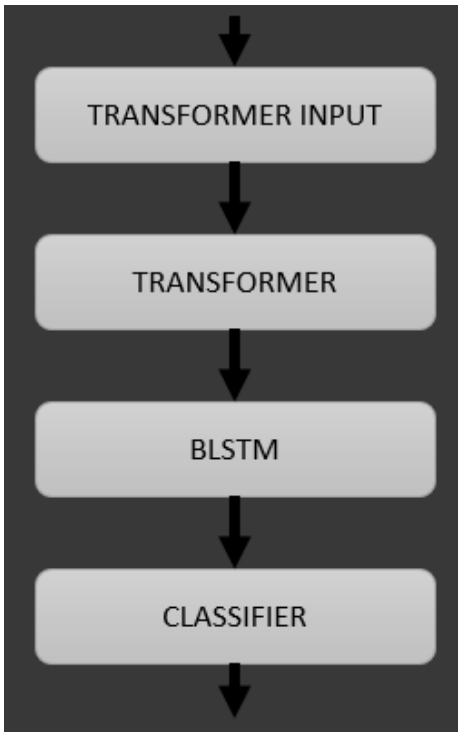


Figure 5: The architecture of the base model without the predicate Embedding.

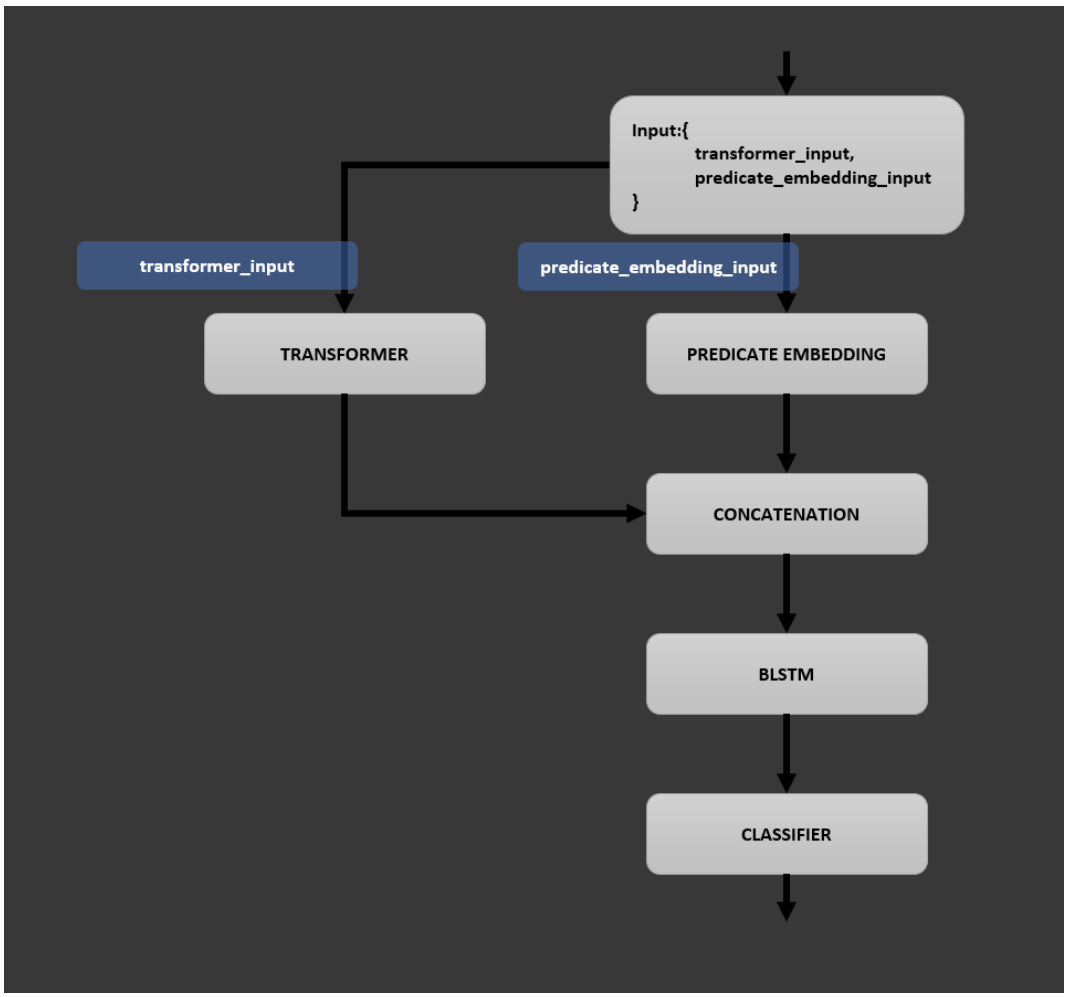


Figure 6: The architecture of the base model.

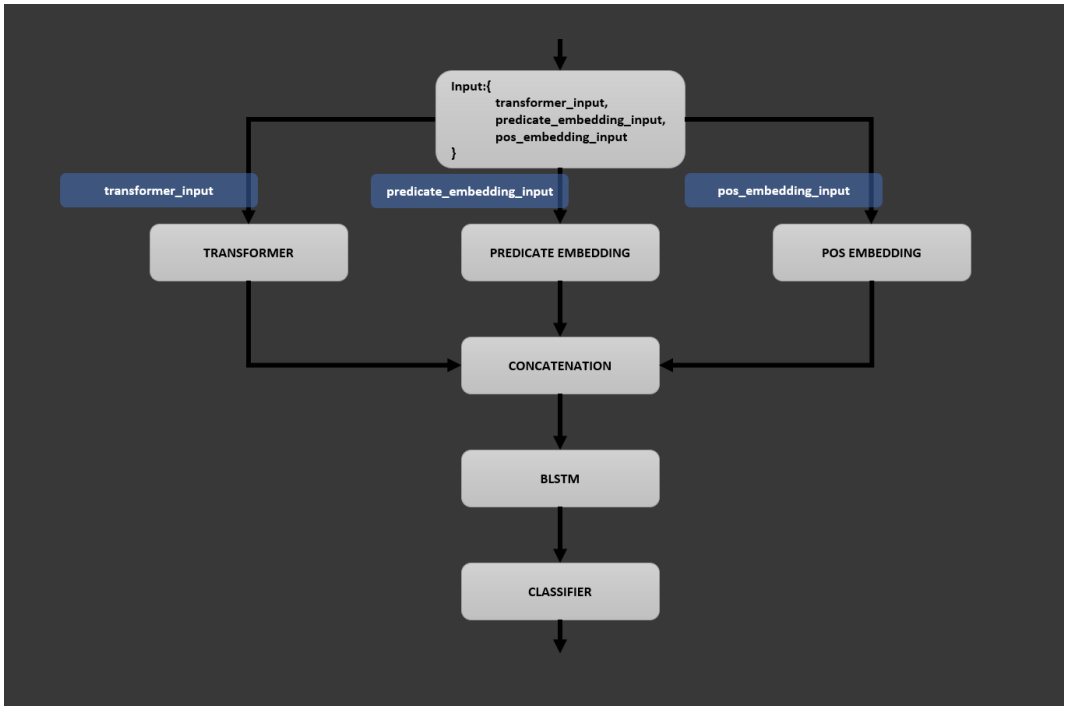


Figure 7: The architecture of the base model plus the POS embedding.

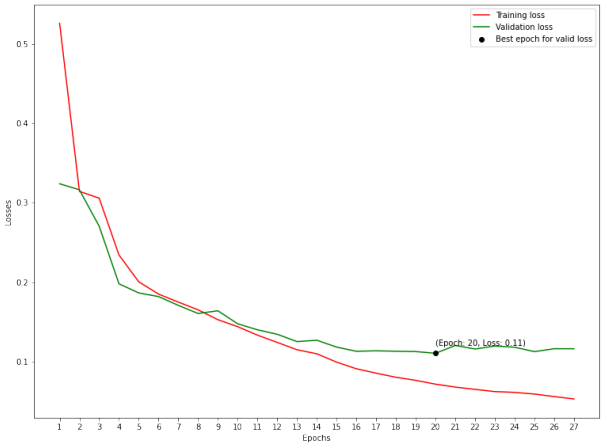


Figure 8: The loss function of the base model.

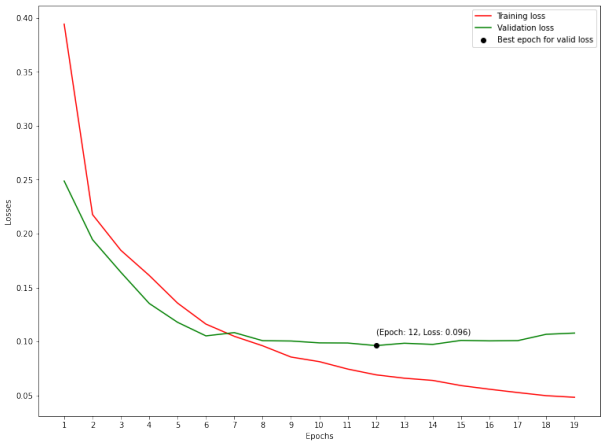


Figure 9: The loss function of the base model plus POS embedding.

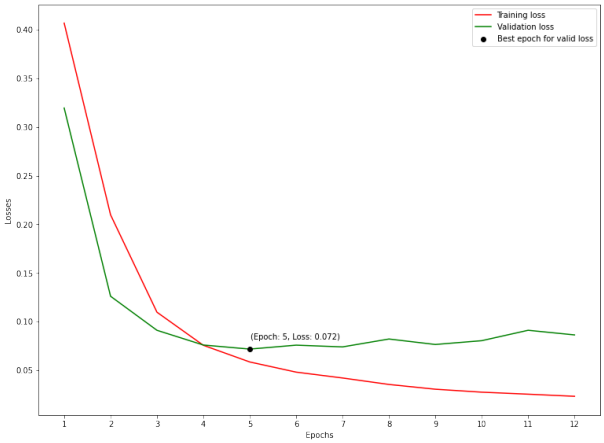


Figure 10: The loss function of the base model plus POS embedding exploiting VerbAtlas.

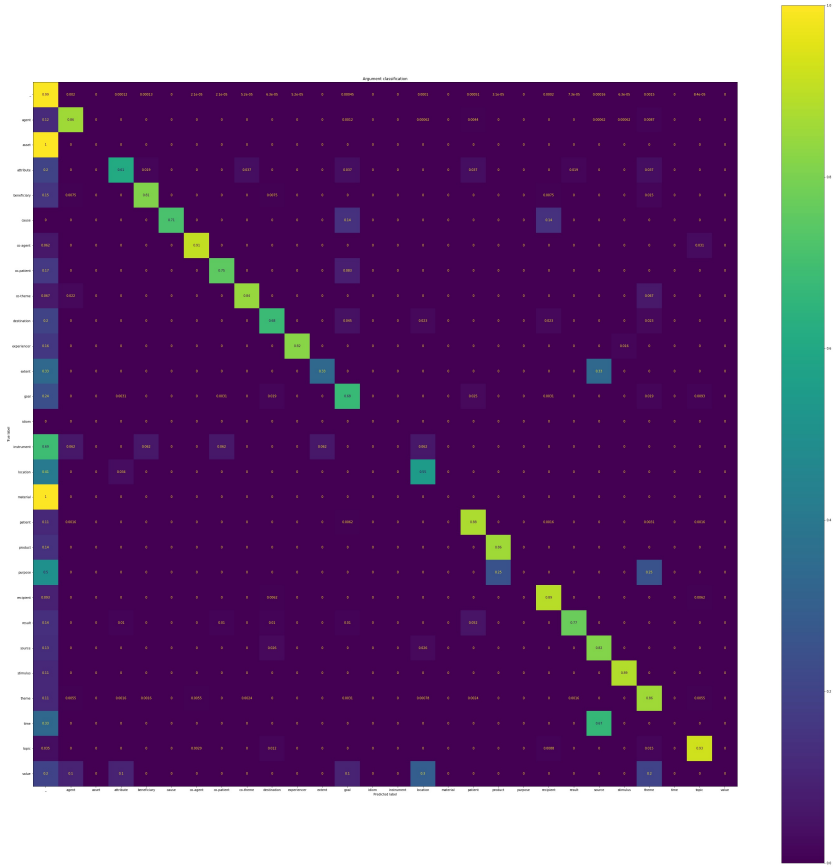


Figure 11: The confusion matrix for *Argument Classification* for the best checkpoint of the model base plus POS Embedding exploiting VerbAtlas.

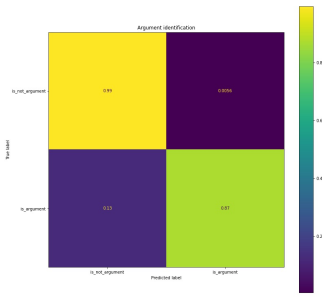


Figure 12: The confusion matrix for *Argument Identification* for the best checkpoint of the model base plus POS Embedding exploiting VerbAtlas.

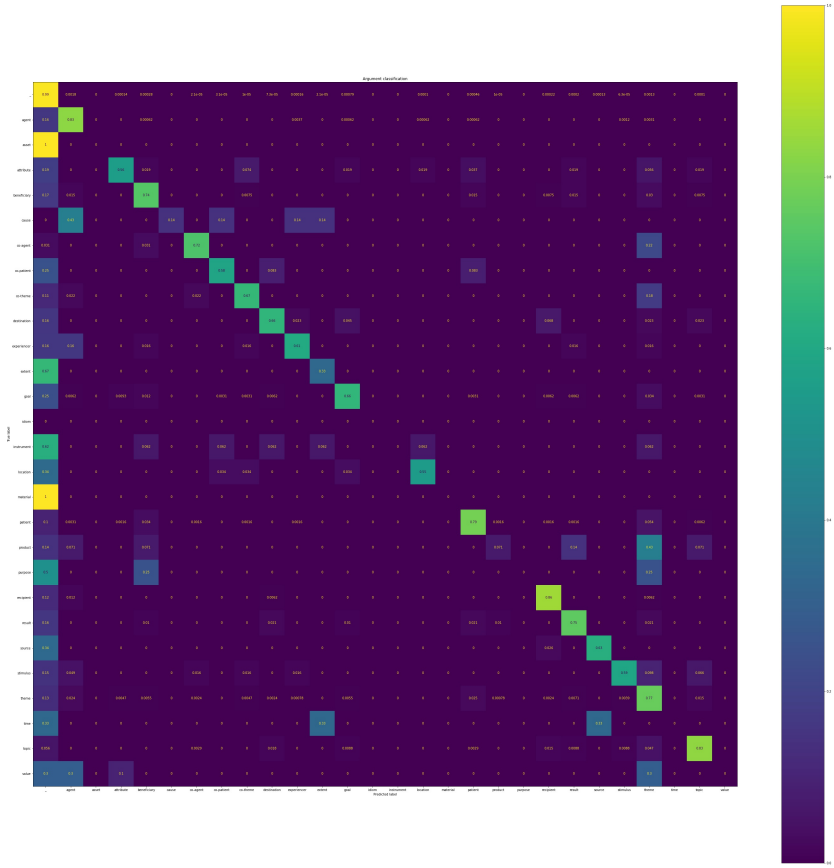


Figure 13: The confusion matrix for *Argument Classification* for the best checkpoint of the model base plus POS Embedding.

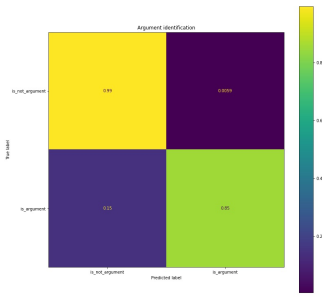


Figure 14: The confusion matrix for *Argument Identification* for the best checkpoint of the model base plus POS Embedding.

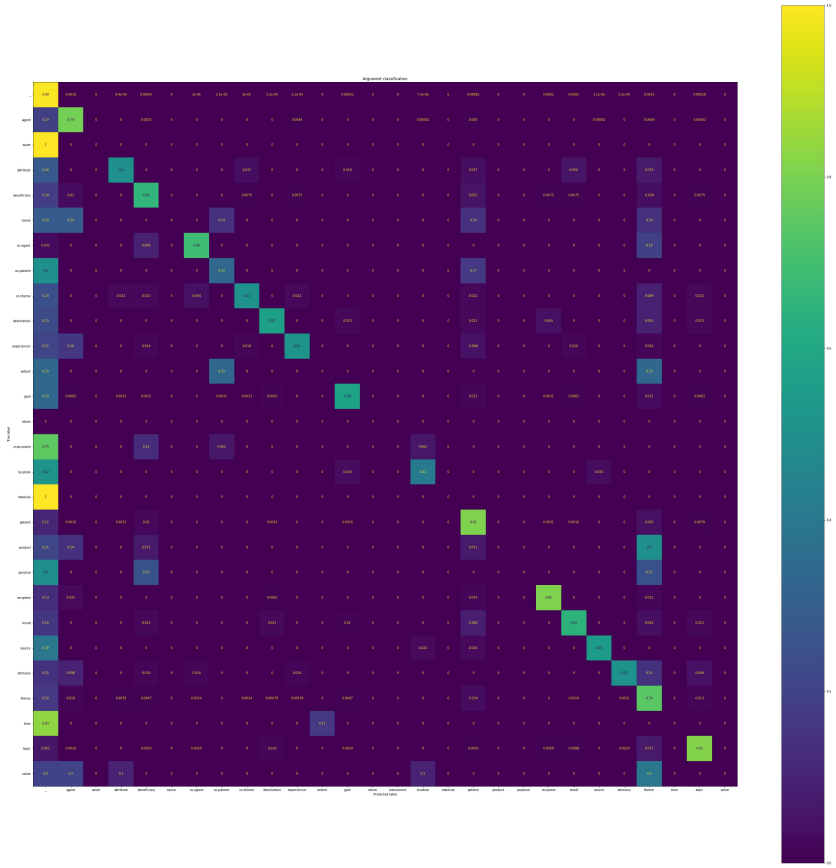


Figure 15: The confusion matrix for *Argument Classification* for the best checkpoint of the model base.

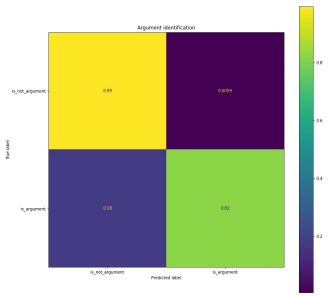


Figure 16: The confusion matrix for *Argument Identification* for the best checkpoint of the model base.