NLP Homework 2 - Report

Andrea Caciolai (1762906)
Master Degree in Artificial Intelligence and Robotics
caciolai.1762906@studenti.uniroma1.it

1 Introduction

One of the most exciting areas seeing recent development in Natural Language Processing (NLP) is concerned with *semantics*, i.e. going beyond syntax to capture the underlying meaning of language.

In this context, *Semantic Role Labeling* (SRL) is the central task of an approach to semantic representation called *predicate-argument semantics* (Eisenstein, Sec. 13.1), providing representations that have been shown to be beneficial in many NLP applications (Marcheggiani et al., 2017). In particular, the goal of SRL is to identify and label the arguments of a given predicate in a sentence, belonging to a *predicate frame*, according to a set of predefined relations (e.g., "who" did "what" to "whom"), called semantic roles. These relations embody the logical relations that link arguments to their predicates and provide a layer of abstraction beyond simple syntactic dependency relations (Roth and Lapata, 2016). For instance, consider the sentences "Alice taught Bob algebra" and "Alice gave Bob a book". The predicate Teach has Alice and Bob as arguments with the roles `Donor` and `Recipient`, while the predicate Give has them with different roles, such as `Teacher` and `Student`. This difference would be flattened in a merely syntactic dependency analysis since both would produce a subject-object relation between Alice and Bob.

Practically, the task of SRL consists in "filling" the semantic roles for a given predicate in a sentence with the appropriate words (Eisenstein, Sec. 13.2). Traditionally the task is considered as divided into four subtasks: *predicate identification* and *disambiguation*, *argument identification* and *classification*. For this homework, we were required to implement an end-to-end model to perform the last two subtasks, with the option of extending it further to also include the first two. In the following, I report, explain and motivate the work I have done in solving this task.

2 Model

In this section I describe the (final version of the) model I used to solve the tasks of argument identification and classification; as I clarify below, the model is very easily extended for solving also predicate disambiguation, therefore my model solves the last three subtasks in the SRL pipeline, and in this report I focus on the last two.

SRL can be treated as a *classification* problem, more specifically a *sequence labeling* problem, the task of assigning to each word in a sentence a label from a set $\mathcal{Y}$ with respect to each predicate, that can be formalized as follows:

$$\hat{y}_{(i,j)} = \arg\max_{y \in \mathcal{Y}} \psi(\mathbf{w}, y, i, j, \rho, \tau) \qquad (1)$$

where (i, j) indicates the *span* of a phrasal constituent, $\mathbf{w}$ is the representation of the sentence as a sequence of embedded words, ρ is the index of the predicate in the sentnece, τ encodes the syntactic structure of the sentence (Eisenstein, Sec. 13.2) and for this homework $\mathcal{Y} = \{\texttt{NULL}, \texttt{Agent}, \texttt{Theme}, \dots\}$ are the labels (semantic roles) taken from VerbAtlas (Di Fabio et al., 2019).

2.1 Model Architecture

The first and last arguments disappear from (1) if one performs *dependency-based* and *syntax-agnostic* SRL, as I did, following (Marcheggiani et al., 2017) and (Shi and Lin, 2019). In fact, my model consists of a series of *embedding* layers (Section 2.2, Section 2.3) producing an embedded sequence that is fed to a stack of BiLSTMs (Section 2.4) that is finally used by a role classifier (Section 2.5) to predict the semantic roles. My

formulation of the model thus reduces to

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^n} \psi(\mathbf{w}, \mathbf{y}, \rho), \tag{2}$$

with sentences with multiple predicates being "duplicated" so that the model considers only one predicate for each sentence, and assigns labels (semantic roles) with respect to that predicate. This duplication is justified since it is expected that words will have multiple, different roles for different predicates in a multi-predicate sentence. Summarizing, the model that implements (2) can be formalized as

$$\begin{aligned} z^{(1)} &= BiLSTM(X(\mathbf{w}, \rho)) \\ z^{(k)} &= BiLSTM(z^{(k-1)}) \\ \psi(\mathbf{w}, \mathbf{y}, \rho) &= \mathbf{\Theta}^{(\mathbf{y})} z^{(k)} \end{aligned} \tag{3}$$

where X is an embedded representation of sentence $\mathbf{w}$, considering predicate ρ, and $\mathbf{\Theta}^{(\mathbf{y})}$ is the linear classifier. See Fig. 1 for a graphical representation of the model architecture.

2.2 Word Representation

I represent each word w in a given sentence, seen as a sequence of words, as the concatenation of four vectors, similarly to (Marcheggiani et al., 2017). Given a pre-trained word embedding $\mathbf{x}^{we} \in \mathbb{R}^{d_w}$, a trainable lemma embedding $\mathbf{x}^{le} \in \mathbb{R}^{d_l}$, a trainable part-of-speech (POS) tag embedding $\mathbf{x}^{pos} \in \mathbb{R}^{d_p}$, a *predicate frame* embedding $\mathbf{x}^{pf} \in \mathbb{R}^{d_f}$, the word representation is given by $\mathbf{x} = \mathbf{x}^{we} \circ \mathbf{x}^{le} \circ \mathbf{x}^{pos} \circ \mathbf{x}^{pf}$ where $\circ$ represents the concatenation operator.

I used pre-trained word-embeddings (FastText, see Table 2 for details) as a baseline and then increased the representational complexity by including more information, for the following reasons. Words that represent the same predicate but conjugated differently will have different word embeddings but same lemma embedding, therefore making the model more robust in that regard. Then, POS tag embeddings notoriously improve the performance of downstream NLP tasks, and SRL is one of those since words tagged as e.g. a noun will likely be an argument of a predicate. Last but not least, information on the predicate frame is crucial to classify correctly the semantic roles, since the roles are specific to each frame, and in fact it is the one information I have always used along with word embedding.

Note that the information on the predicate frame is only available when solving the SRL task from argument classification onwards, while it must be predicted by the model itself (and then returned and also relayed downstream) when solving also predicate disambiguation. The upstream "submodel" responsible to do so, therefore, receives word representations that lack this information.

2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) is one of the most recent language representation models, leveraging the self-attention mechanisms and Transformer architecture popularized by (Vaswani et al., 2017). It can be used as a pre-trained model to obtain *contextualized* embeddings that have been shown to be well separated based on the various senses of a given word, therefore providing a *semantic vector representation* for a whole sentence (Coenen et al., 2019).

I included BERT in my model since the sentence representation as a simple sequence of word embeddings is only local, clearly unable to capture any contextual information, and more notably lacks any semantic information whatsoever. In fact, one may think that the first issue is solved when introducing a downstream BiLSTM as I do, but several sources including (Shi and Lin, 2019) report that such models can have great benefit by leveraging BERT contextual information and even yield state-of-the-art performance on a variety of benchmarks for the SRL task. For this reasons, I include a BERT embedding of the sentence in the final representation, that has thus the form $\mathbf{X} = \mathbf{X}^{bert} \circ \mathbf{X}^{word}$ where $X^{bert} \in \mathbb{R}^{n \times d_b}$ is the BERT embedding of the sentence and $X^{word} = \begin{pmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_n \end{pmatrix} \in \mathbb{R}^{n \times (d_w + d_l + d_p + d_f)}$ is a matrix collecting the word representations of the n words that compose the sentence.

2.4 BiLSTM

In my model, the representation $\mathbf{X}$ of a sentence $\mathbf{w}$ described above is linearly projected onto a hidden space $\mathbb{R}^{n \times d_h}$ in which a stack of BiLSTMs work to extract a meaningful nonlinear hidden representation $\mathbf{h}_t \in \mathbb{R}^{d_h}$ for each word w_t in the sentence. The BiLSTM (Bidirectional Long Short-Term Memory) makes use of two LSTM networks (Hochreiter and Schmidhuber, 1997) to capture long-range dependencies in both directions in a sequence. I decided to employ BiLSTM instead of alternatives like a Transformer model, that has been shown to be equally effective and also faster

to train[1], because my pipeline already contains a Transfomer, albeit pre-trained, and most importantly (as mentioned above) the combination of the two has been shown to be effective, retaining the strengths of both models (Shi and Lin, 2019). The module responsible to predict the predicate frames is a BiLSTM as well, that is fed smaller embeddings, as explained in Section 2.2.

2.5 Role Classifier

Following (Marcheggiani et al., 2017) and (Shi and Lin, 2019), for the final prediction of the roles the hidden representation $\mathbf{h}_p$ of the sentence predicate is concatenated to each hidden representation $\mathbf{h}_t$ of each word and then fed into a simple linear classifier over the label set $\mathcal{Y}^n$.

3 Results

Here I report the results I attained with the model described above, implemented in PyTorch as per requirements of the homework.

3.1 Experimental Setup

The experiments and the training have been conducted on Google Colab, that freely provides machines with an *Intel Xeon* CPU, approximately $12GB$ of RAM and a *Nvidia K80* GPU with $12GB$ of dedicated memory, allowing intense parallelization and thus faster training.

3.2 Training

The model is trained by optimizing its parameters $\boldsymbol{\theta}$ to minimize the *cross-entropy* loss

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\hat{\mathbf{y}} \sim \hat{\mathbf{p}}_{data}} \log \mathbf{p}_{model} \tag{4}$$

with $p_{model,i} = Softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$ and z_i unnormalized log probabilities output by the model for each class i (Goodfellow et al., 2016, Chap. 6). The distribution $\hat{\mathbf{p}}_{data}$ for the gold labels $\hat{\mathbf{y}}$ is given by the annotated dataset consisting of $40k$ sentences, split as to have $39k$ examples for training and $1k$ examples for validation, provided for the homework. Note that the loss function is extended to include also the task of predicate disambiguation as follows

$$J(\boldsymbol{\theta}) = (1-\alpha)J_{args}(\boldsymbol{\theta}) + \alpha J_{preds}(\boldsymbol{\theta}) \tag{5}$$

[1]Since self-attention layers are faster than recurrent layers when the sequence length n is less than the representation dimensionality d (Vaswani et al., 2017), which is most often the case and would be the case also in this project.

with J_{args}, J_{preds} defined as (4) but with $\mathbf{y}$ being either argument classes or predicate classes, and $\alpha = \frac{\#\text{argument classification classes}}{\#\text{predicate disambiguation classes}}$ being the linear interpolation coefficient, designed to give balanced importance to both tasks in the optimization.

The model is trained to minimize the loss function (4) (or (5)) with the *Adam* optimizer. The model is regulated with early stopping to avoid overfitting, by monitoring the loss on the validation data, for a maximum of 50 epochs, usually converging in 20 and 35 epochs for the main and "extra" task, respectively (see Fig. 5 and Fig. 6). The model is further regularized by using dropout, shown to avoid co-adaptation and to be one of the few effective regularizers of RNNs (Srivastava et al., 2014) (Pascanu et al., 2012). Batch normalization, shown to improve the performances of NNs (Ioffe and Szegedy, 2015), is also used.

A selection of the model hyperparameters is fine-tuned to obtain the best possible performance, using a basic self-made grid search on a certain range for each parameter. Specifically, I tuned the trainable embedding dimensions, the hidden dimension, the number of layers of the BiLSTMs, the learning rate and the dropout; see Appendix A.2 for the details.

3.3 Performance

The model performance is tested on a test dataset containing $2k$ sentences, and evaluated in terms of the macro-F1 score for both the tasks I implemented, since the number of classes is large and following very unbalanced distributions, with the `NULL` label for both being largely predominant (see Fig. 4 and Fig. 3) as expected since most words in a sentence are not predicates, and given a predicate they are not its arguments (see Fig. 2). The final, fine-tuned versions of the base and extended model scored 0.81 and 0.77 of macro-F1 (using *sklearn* metrics), respectively, on the test dataset. Given the incremental approach I employed, an *ablation study* is reported in Appendix A.3 to investigate the contribution of each addition to the baseline model.

4 Conclusions

The model I presented is structurally simple and able to achieve a good F1-score on the SRL tasks of predicate disambiguation, argument identification and classification, and could be further improved in the future by including also syntactic information like the dependency structure of a sentence.

A Additional Material

A.1 Figures

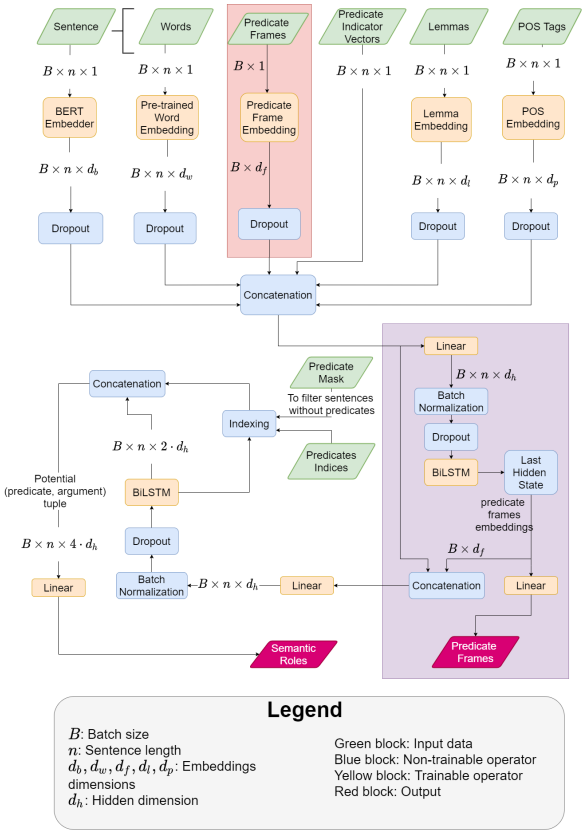


Figure 1: Graphical representation of the model. If performing also predicate disambiguation, the predicate frames (red section) are not available, and the model must predict them by itself (purple section). Otherwise, the first concatenation leads directly to the linear projection block.

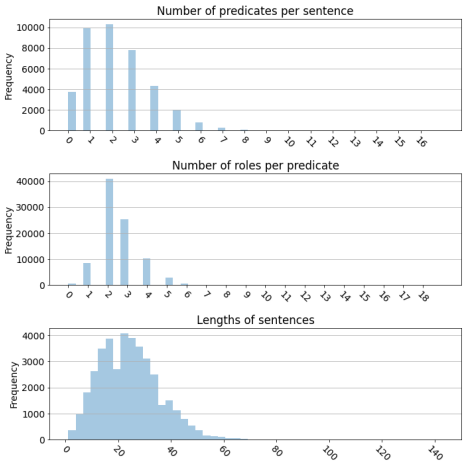
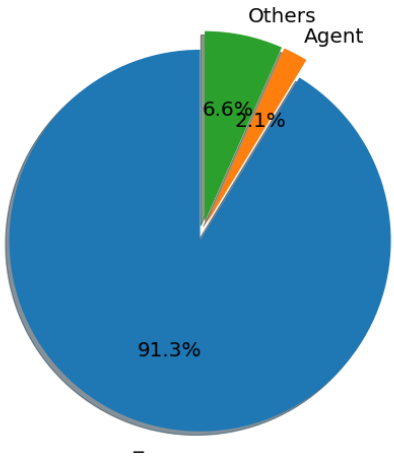
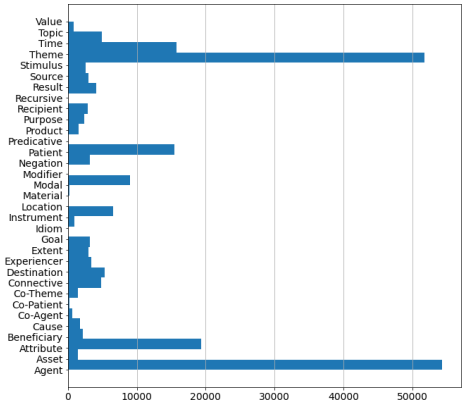


Figure 2: Distribution of predicates and arguments in relation to the length of the sentences.

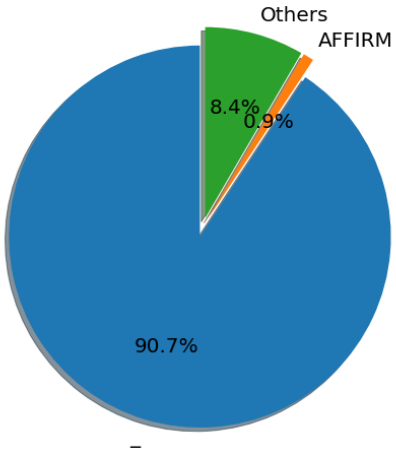


(a) With the NULL label "_"

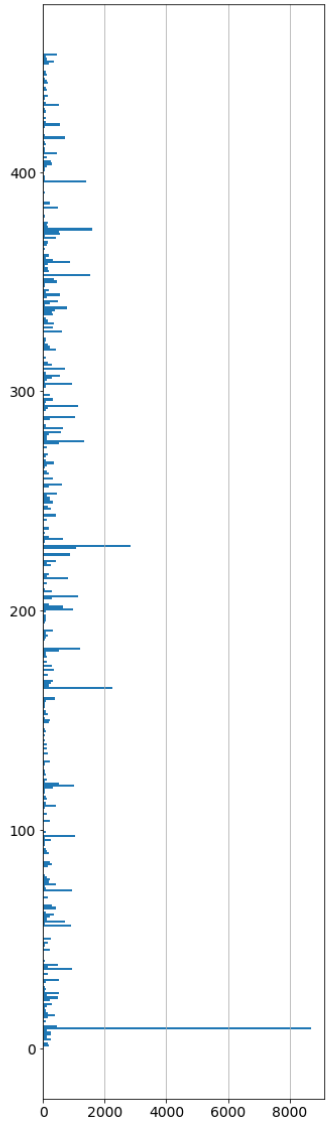


(b) Without the NULL label "_"

Figure 3: Distribution of the labels for argument classification. We can see that even without the NULL label they are still unbalanced.

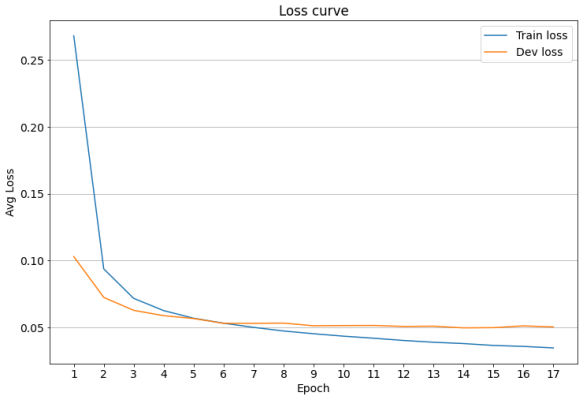


(a) With the NULL label "_"

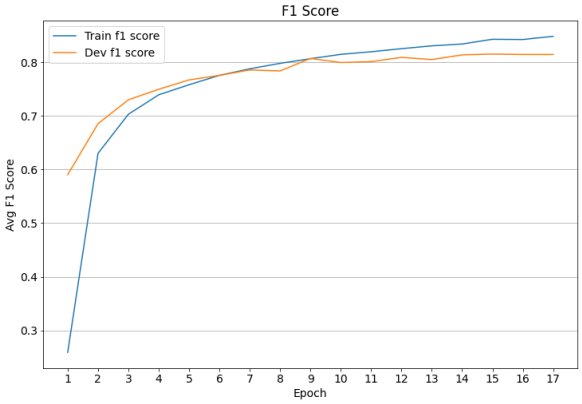


(b) Without the NULL label "_". Explicit labels not reported to avoid cluttering.

Figure 4: Distribution of the labels for predicate disambiguation. We can see that even without the NULL label they are still unbalanced.



(a) Loss curve



(b) F1-Score

Figure 5: Loss and F1 score on the train and dev set during training for the model solving argument identification and classification. The F1 score includes both since a prediction "_" for a word that should be an argument, and vice-versa, is considered a wrong prediction. Notice that by the 14-th epoch the dev loss stops increasing for 3 successive epochs and thus training is stopped early.

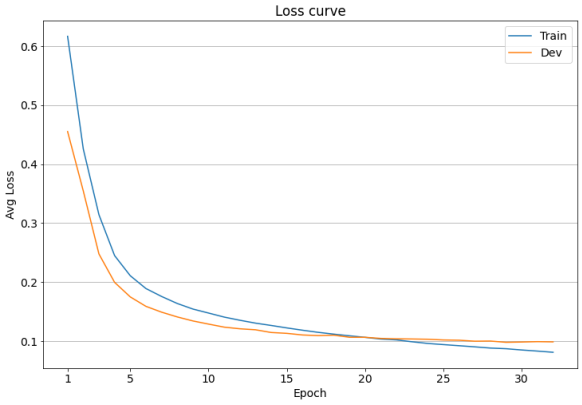
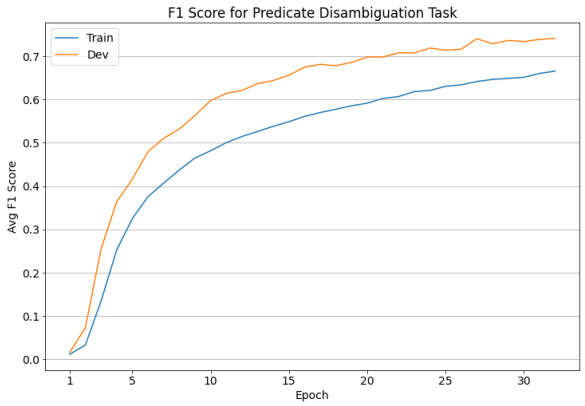
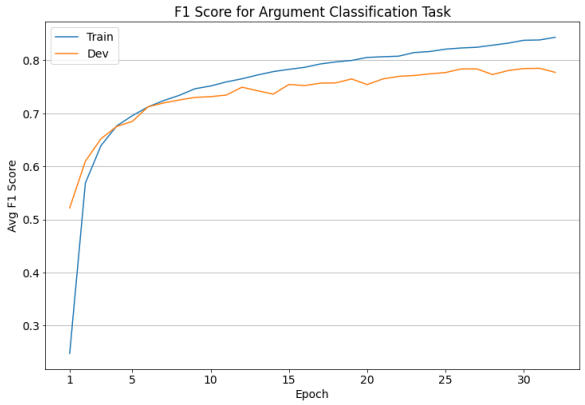


Figure 6: Loss curve during training for the model solving also predicate disambiguation. Notice that by the 29th epoch the dev loss stops increasing for 3 successive epochs and thus training is stopped early.



(a) F1-Score for predicate disambiguation



(b) F1-Score for argument classification

Figure 7: F1 score on the train and dev set during training for the model solving also predicate disambiguation. Also the F1 score per predicate disambiguation is reported.

A.2 Hyperparameters

Here I report the hyperparameters used for training the final model (see Table 2), and also describe the fine-tuning process. The parameters combination performing the best on the test dataset was retained (see Table 1).

Notice that the model chose the largest hidden dimension d_h, but on the other hand chose the largest only for d_f and not for d_l and d_p. I believe this happened because the predicate frames contribute greatly to the performance, so a more expressive embedding is better, while even though the POS tags contribute in a considerate amount (as shown in Table 3), they are in a small number, therefore, a smaller embedding suffices; on the other hand, even though the lemmas are much more ($\approx 30k$ different lemmas in the training set) they do not contribute as much, so a tradeoff was made.

Parameter	Tested value	Best value
Nr. of BiLSTM layers	$\{1, 2, 3\}$	2
d_h	$\{32, 128, 512\}$	512
d_f	$\{32, 64, 128\}$	128
d_l	$\{32, 64, 128\}$	64
d_p	$\{32, 64, 128\}$	32
Dropout	$\{0.1, 0.3, 0.5\}$	0.5
Learning rate	$\{10^{-4}, 5 \cdot 10^{-4}, 10^{-3}\}$	10^{-3}

Table 1: Fine-tuning of a selection of the hyperparameters.

Parameter	Value
Nr. of BiLSTM layers	2
d_w	300 (FasText Wiki-1M)
d_f	128
d_l	64
d_p	32
d_b	768 (BERT-base-cased)
B	128
Dropout	0.5
Learning rate	10^{-3}

Table 2: The final table of hyperparameters, see Fig. 1 for a legend of the symbols (used for brevity). The dimensions of the pre-trained embeddings are prescribed. I used FastText Wiki-1M as pre-trained word embedding since it worked well for me in Homework 1, and I verified that also in this Homework it provides a good tradeoff between complexity and performance. I used BERT-base-cased since the large version seemed overkill, and the casing might be relevant to distinguish roles that usually belong to human beings (e.g. a proper noun for the role `Teacher` as in the example sentence in Section 1).

A.3 Ablation Study

Here I report the results of the *ablation study* I performed.

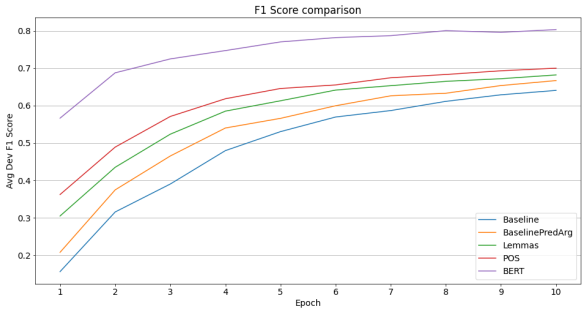


Figure 8: The models have all been trained on the argument identification and classification tasks for 10 epochs, to ensure a homogeneous comparison.

Variant	Test F1-score
Baseline	0.60
Baseline with joint (predicate, argument) classification	0.63
Lemma Embeddings	0.64
POS tags Embeddings	0.67
BERT Embeddings	0.77
Fine-tuning	0.81

Table 3: Comparison of the performances of the incrementally more complex models I developed. The table shows models by increasing complexity, i.e. models below in the table include the features of the models above. The performance metric used is the macro-F1 computed by *sklearn* on the test set to homogeneously compare them. It is evident that adding the BERT embeddings helped the performances significantly, as I expected given the considerations done in Section 2.3.

References

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of bert.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. VerbAtlas: a novel large-scale verbal semantic resource and its application to semantic role labeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 627–637, Hong Kong, China. Association for Computational Linguistics.

Jacob Eisenstein. *Introduction to natural language processing*. The MIT Press.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks.

Michael Roth and Mirella Lapata. 2016. Neural semantic role labeling with dependency path embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1192–1202, Berlin, Germany. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.