Report homework NLP 2019/2020: Semantic Role Labeling

Simone Antonelli
Department of Computer Science, Sapienza University of Rome
antonelli.1753685@studenti.uniroma1.it

1 Introduction

A fundamental task in Natural Language Processing is the task of Semantic Role Labeling which extract the predicate-argument structure of a sentence, determining "who did what to whom", "when", "where" and "how". The whole SRL process is divided into four steps:

1. **predicate identification**: which consists in identifying the predicates in a sentence;
2. **predicate disambiguation**: which consists in determining the type of the identified predicates;
3. **argument identification**: which consists in identifying the arguments in a sentence of a specific predicate;
4. **argument classification**: which consists in determining the role of the arguments in relation to a specific predicate.

For the purpose of the homework, the mandatory tasks are argument identification and argument classification, while the other two are optional.
In this work two models are presented: one which deals with the tasks argument identification and argument classification, and the other which deals with the tasks predicate disambiguation, argument identification and argument classification; from now on, in the report, these two models will be respectively named M_1 and M_2.
Since both the tasks argument classification and predicate disambiguation can be seen as sequence labeling tasks, the neural model with which to face these tasks is the bidirectional LSTM that can capture syntactic information leveraging the context created by its states. Moreover, the two models use a pretrained BERT model to get the embedding representation of each word of a sentence during the preprocessing of the dataset.
In the next sections are described the structures of the models, the encoding of the input both in M_1 and M_2 with their training and validation phases and an analysis on the experimental outcomes obtained improving increasingly the models.

2 Neural architecture

The architectures of M_1 and M_2 are very similar, in particular, M_2 includes M_1 to face the tasks of argument identification and argument classification, but encodes the sentences in a different way.
M_2 is composed by two stacked bidirectional LSTM (Long Short-Term Memory), the former LSTM dealing with predicate disambiguation task, while the latter LSTM, leveraging the results of the first, faces the argument identification and classifications tasks. Instead, M_1 is composed of only a bidirectional LSTM which faces the tasks of argument identification and argument classification.
Both the models use a pretrained Bert model during the preprocessing of the dataset to get the embedding representation of the sentence, while to represents the other syntactic information (Marcheggiani et al., 2017) they use different embedding layers.
The flow of the developed models can be seen in fig. 1 and fig. 2.

2.1 BERT model

Taking inspiration from (Shi and Lin, 2019), during the preprocessing of the dataset, instead of using a pretrained word embedding, a pretrained BERT model is used to get the embedding representation of a sentence, which has shown impressive gains in many of natural language tasks among which there are the sequence labeling tasks. BERT, which stands for Bidirectional Encoder Representations from Transformers, is a neural model that makes use of Transformer, an attention mechanism that learns contextual relations between words and/or

sub-words in a text.
There are many pretrained models that differ in the number of parameters and for this work, I adopted the simplest pretrained BERT model (*'bert-base-cased'*) which has 12 layers, a hidden size equals to 768, and the number of self-attention heads equals to 12, for a total number of parameters equals to 110M (as mentioned in (Devlin et al., 2018));

2.2 Embedding layers

With the aim of giving the model more information as possible, the dataset provides much syntactic information. From the set of information, those used are pos tags, predicate frames and lemmas (Marcheggiani et al., 2017); these are exploited in different ways in the two models M_1 and M_2, but the thing that have in common is that they are represented using an embedding layer.
After getting the contextual embedding representation of the sentence using BERT model, depending on the model, to the sentence is concatenated the information obtained from the embedding layers.

- M_1: In the following model the sentence is simply represented by the concatenation of all the selected information; so, let $x_i = x_i^{\text{bert}} \circ x_i^{\text{pred\_ind}} \circ x_i^{\text{pos}} \circ x_i^{\text{preds}} \circ x_i^{\text{lemmas}}$ be the representation of the token i of the sentence where $\circ$ is the concatenation operator; then, x_i^{bert} represents the contextual embedding of the token i, $x_i^{\text{pred\_ind}}$ represents the predicate indicator of the token i which is 1 if x_i is a predicate 0 otherwise, x_i^{pos} represents the embedding of the pos tag of the token i, x_i^{preds} represents the embedding of the predicate frame of the token i and x_i^{lemmas} represents the lemma embedding of the token i;

- M_2: since this model is composed of two bidirectional LSTM where the latter leverages the output of the former, a different representation is adopted. To the first LSTM which faces predicate disambiguation task, the sentence is fed to it with only the information useful to disambiguate the predicates, so $x_i^{\text{LSTM}} = x_i^{\text{bert}} \circ x_i^{\text{pred\_ind}} \circ x_i^{\text{pos}} \circ x_i^{\text{lemmas}}$, but to add more relevance to the token that represents the predicate, x_i^{lemmas} is the embedding of the lemma of the token i only if in position i there is a predicate, while about the pos tag embedding is better to have information about all the tokens because to disambiguate a predicate is useful also know what part of speech the other tokens play in the sentence. The second LSTM that composes M_2 leverages the output of the previous LSTM to face the tasks of argument identification and argument classification. So, the representation of each token of a sentence is the following $x_i = x_i^{\text{bert}} \circ x_i^{\text{LSTM}} \circ x_i^{\text{lemmas}}$ where in this case the embeddings of the lemmas are all used. Note that the pos tag embeddings are not used since in x_i^{LSTM} there is already the information about them.

2.3 BiLSTM

Long short-term memories (LSTM) are a variant of the *Recurrent Neural Networks* that use memory cells to propagate information through the sequence without applying non-linearities. The reason for using the bidirectional LSTM is to create the context of each word in the sentence using two different LSTM: one which computes the left context of each token in a sentence and a second LSTM which reads the same sequence in a reverse way to create the right context of the tokens; the two contexts are concatenated to obtain the representation of each token in the sentence.
In both the models M_1 and M_2, before to do role classification, the hidden state of the predicate returned by the LSTM is concatenated to the hidden state of each token of the sentence and then fed it into a fully connected layer to do the classification; this is done to add more dependency from the predicate to each token of the sentence (Shi and Lin, 2019).

3 Experiment

The models are built increasingly starting from using only the word embedding to represent the sentence until using almost all the syntactic information given in the dataset. The performances that the model achieves during the improvements are shown in table 3.

3.1 Encoding of the input

To represent the sentences as explained in Embedding layers, some vocabularies are made to represent all the information about the pos tags, the lemmas, the predicates and the roles (this is necessary to evaluate the performances of the model). After verifying that all the pos tags and roles that are in the validation and the test dataset are also

in the training dataset, their vocabularies are built using only the information in the provided dataset; while, about the vocabulary of the predicate frames, I used VerbAtlas (Di Fabio et al., 2019) which provides all the possible frames avoiding the use of '⟨unk⟩' token, which is instead necessary to build the vocabulary of the lemmas.
Due to the fact that in the dataset there are sentences without predicates, these are handled as information of relevance but since in the dataset the dictionary of the roles for these sentences is empty, it is replaced with an array filled with the null token; in this way, no information is lost but rather it helps the model to identify in a better way which token will be predicted as a null token.
In both the models, the BiLSTM which predicts the roles takes as input a sentence with only one predicate in order to predict only the roles that each token of a sentence assume considering that specific predicate and to avoid missing predictions since there can be a token that has a different role for two distinct predicates. So, for the sentences with many predicates, the model in its forward step split them and assign a predicate to each of these duplicated sentences.

3.2 Training phase

The models are trained using Adam optimizer with the parameters listed in table 1 for the model M_1, while the parameters listed in table 2 are those for the model M_2. Furthermore, different loss functions are used for the models: since M_2 must face two different tasks, a linear combination of two different losses is used, one cross-entropy function for the task of predicate disambiguation and another cross-entropy function for the task of role classification. About this linear combination, I decided to give more relevance to the loss of the role classification task since it is the main task of the SRL pipeline and also because is more difficult due to the fact that for predicate disambiguation, the position of each predicate is already known, while for the role classification task is not the same, so the loss for predicate disambiguation task has a weight of 0.4 and the loss for role classification task has a weight of 0.6. While about M_1, only a cross-entropy function is used for the role classification.
To improve the models' ability to generalize, and to avoid overfitting, some regularizers are introduced. One of them is the dropout layer which is applied on the whole word representation, on the input and on the output of the BiLSTM that takes care about role classification task; moreover, it is applied also on the output of the BiLSTM which takes care about predicate disambiguation task. Another regularizer is the gradient clipping, used to avoid exploding gradients which negatively impacts the performances of the LSTMs.
The last introduced regularizer is a simply early stopping with the patience of 5 epochs and a change of $1e$-4 which breaks the training loop if the f1-score does not improve at least $1e$-4 for 5 consecutive epochs, and restore the weights of the best model, namely the one with highest f1-score.

3.3 Results

The performance of the final models on the given test dataset is shown in bold in table 3. The shown outcomes are obtained after 17 epochs, as shown in fig. 6, for M_1 where each of these takes about 40 seconds, while 22 epochs are needed to obtain the scores for M_2, as shown in fig. 7, where each epoch takes about 1 minute and 10 seconds; both the training of M_1 and the training of M_2 are stopped restoring the epoch in which the model had the maximum f1-score, as can be seen in fig. 8 and fig. 9. Note that the metric used to evaluate the performances of the model is not the accuracy since the number of possible roles (and also the number of possible predicates in M_2) is very high and the given train dataset is imbalanced (there are roles that do not appear as a label in the training dataset) as can be seen in fig. 3. Moreover, since the tasks are multi-class problems, the adopted metric to evaluate the models is the f1-score.
These results are not the same of the ones obtained with a state-of-the-art model, but they are very close, so the proposed model is a good baseline to face the SRL task; in the section Conclusion is presented an idea to improve these results.

4 Conclusion

The proposed model is composed by only two stacked bidirectional LSTM, nevertheless, it can achieve a very good f1-score leveraging the syntactic information provided by the dataset, but mainly a significant improvement is given by the contextual embedding returned by BERT model. Moreover, for future works, the model can be extended considering the remaining syntactic information in the dataset like dependency heads and dependency relations.

Additional Resources

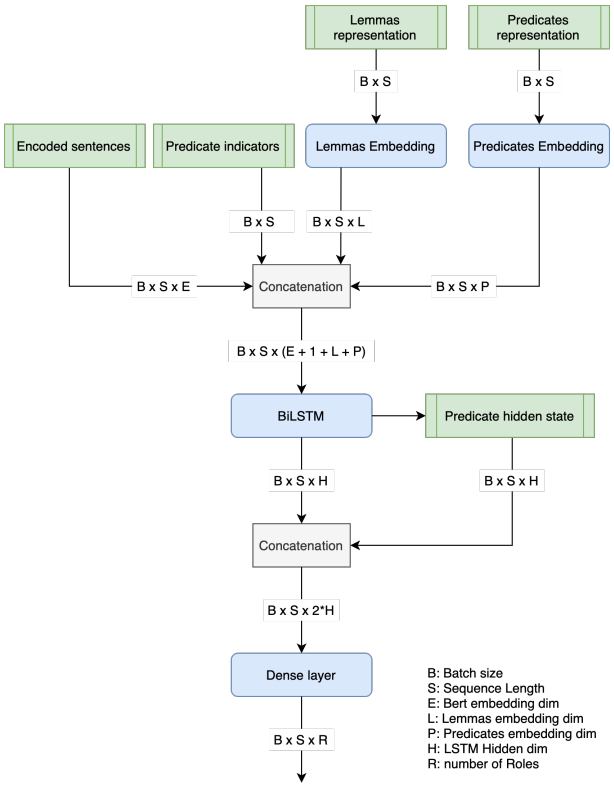

Figure 1: Representation of the flow of the model M_1.

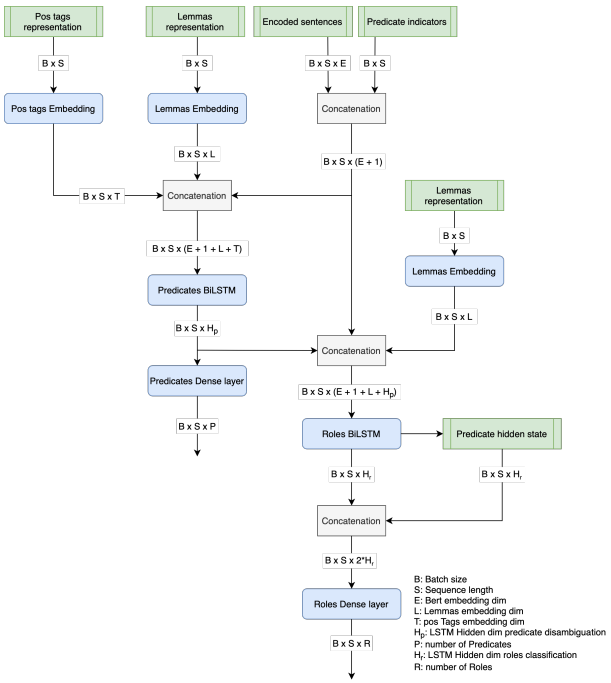

Figure 2: Representation of the flow of the model M_2.

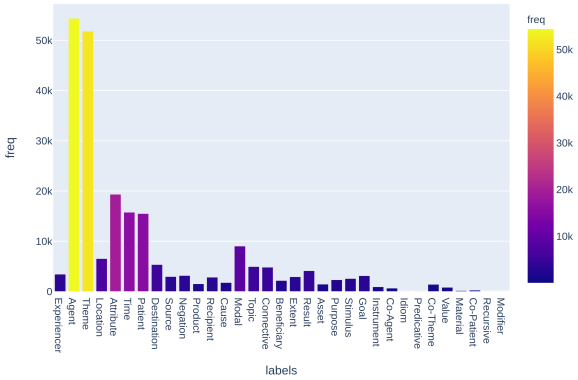

Figure 3: Distribution of the roles in the train dataset. The null token is not considered since it appears a number of times greater than the others roles.

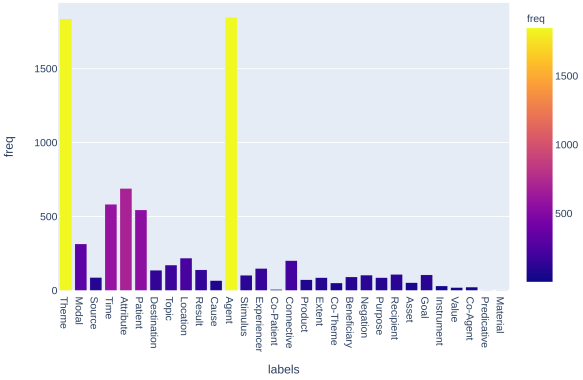

Figure 4: Distribution of the roles in the validation dataset. The null token is not considered since it appears a number of times greater than the others roles.

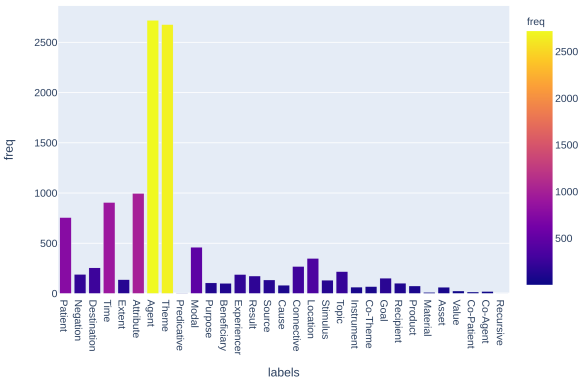

Figure 5: Distribution of the roles in the test dataset. The null token is not considered since it appears a number of times greater than the others roles.

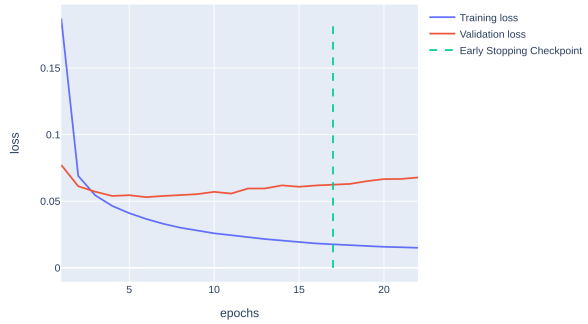

Figure 6: Comparing the losses between the train and the validation datasets during the training of M_1. The dashed line shows the epoch when the model has highest f1-score.

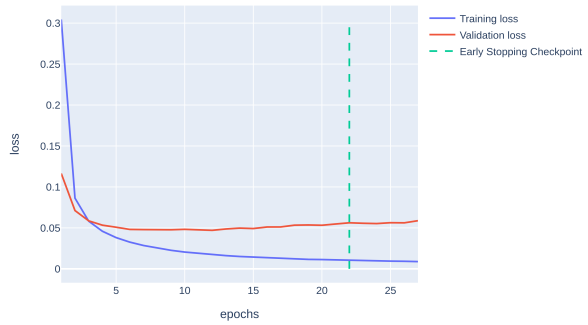

Figure 7: Comparing the losses between the train and the validation datasets during the training of M_2. The dashed line shows the epoch when the model has highest f1-score.

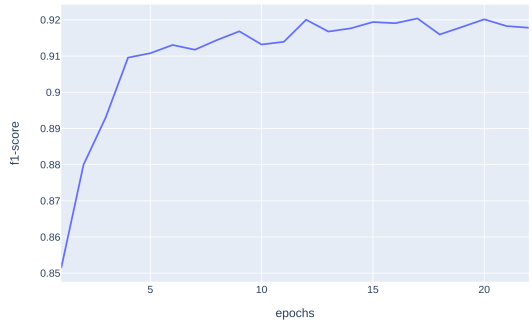

Figure 8: F1-score trend on validation dataset during the training of the model M_1. Its peak is in the same epoch indicated by early stopping in losses figure.

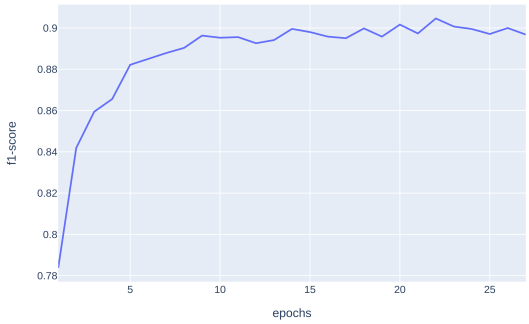

Figure 9: F1-score trend on validation dataset during the training of the model M_2. Its peak is in the same epoch indicated by early stopping in losses figure.

Hyperparameters	
Bert embedding dim	768
Pos tags embedding dim	128
Lemmas embedding dim	200
Predicate embedding dim	200
Hidden dim	512
Batch size	128
Dropout	0.2
Learning rate	$1e$-3
Gradient clipping	2

Table 1: Hyperparameters used for the final model M_1.

Hyperparameters	
Bert embedding dim	768
Pos tags embedding dim	128
Lemmas embedding dim	200
Predicates hidden dim	400
Roles hidden dim	512
Batch size	128
Dropout	0.2
Learning rate	$1e$-3
Gradient clipping	2

Table 2: Hyperparameters used for the final model M_2.

Architecture model	Predicate disambiguation	Argument identification	Argument classification
BiLSTM $(W + P_i)$	-	84.48 %	76.45 %
BiLSTM $(B + P_i)$	-	91.30 %	82.50 %
BiLSTM $(B+P_i+T+L+P)$	-	**93.91 %**	**90.09 %**
Stacked BiLSTM $(B + P_i + O + L)$	**95.17 %**	**93.50 %**	**87.83 %**

Table 3: F1-score performances of the model at each improvement. W: Word embedding; P_i: Predicate indicators; B: Bert embedding; T: Pos tags embedding; L: Lemmas embedding; P: Predicate embedding; O: Predicate embedding returned by the first LSTM.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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.

Diego Marcheggiani, Anton Frolov, and Ivan Titov. 2017. A simple and accurate syntax-agnostic neural model for dependency-based semantic role labeling. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 411–420, Vancouver, Canada. Association for Computational Linguistics.

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