Semantic Role Labeling - NLP Homework 2

Francesco Starna
Sapienza University of Rome
starna.1613660@studenti.uniroma1.it

Abstract

Semantic Role Labeling (SRL) is a crucial step for natural language understanding and is becoming subject to important studies and research. The predominant approach is based on recurrent neural network (RNN), that has been proved to be very effective on large context, however, they suffer from capturing semantic and syntactic long range dependencies. In this paper I present a basic starting model based on long short-term memory (LSTM), and some improvements and variants that achieve high performance and are competitive with recurrent ones.

1 Introduction

SRL is a semantic task of NLP which goal is to determine ”Who did What to Whom, How, When, and Where?”, indicating the event properties between entities in a sentence, such as predicates and arguments. The relationship of the argument with respect to the predicate represents a *role*, that has to be identified and classified, according to VerbAtlas (Fabio et al., 2019) role set. Semantic roles are strictly related to syntax and part-of-speech tagging, and therefore used as features. Each word and its meaning plays a different role in SRL, so pretrained word and contextualized embeddings enriches words representation, behaving differently among the approaches. A first simple one encodes the input sequences in a BiLSTM, while a second more complex focuses on the concept of self attention.

2 Preprocessing

The input data was taken and pre-processed before feeding the neural network. A vocabulary class was built, which contains the encodings of *lemmas* (size=20000), *dependencies* (size=55), *dependency heads*, *postags* (size=40), *predicates* and *roles*, in order to build the corresponding 1-hot feature embeddings and target labels. The lemma encodings are instead used for pretrained word embeddings from global vectors for word representation (GloVe) (Pennington et al., 2014). Global vectors are built according to word co-occurence probabilities on a big text corpora. On the other hand, in order to build the contextualized embeddings, word tokens of each sequence has been fed to a transformer architecture, BERT (Devlin et al., 2019), which uses a ”masked language models” (MLM) to build context representation of words in a sentence. While GloVe embeddings were built easily, BERT ones needed more effort, due to the WordPiece input encodings of the model, that required a substitution of particular symbol tokens, and the reconstruction of the final embedding.

An encoder utility class takes row input samples and encodes each sequence, introducing <pad> tokens (for parallelism purposes) and giving back the encodings corresponding to the vocabularies. The predicate position is essential to let the neural network understand the roles, therefore it is added to the input features as a flag $f \in \{\text{<empty>}, p\}$, where empty means that the current word is not a predicate, while $p \in [0, \text{length(predicates)}]$ represents the corresponding encoded predicate.

In order to fit the attention model, two masks are provided:

Padding. A mask $\boldsymbol{P} \in \mathbb{R}^{s \times s}$ prevents the attention layer to put effort on padding elements. Since the network elaborates batches of data, filled with the <pad> token to normalize the length, such a mask is needed.

Attention. A mask $\boldsymbol{M} \in \mathbb{R}^{sh \times s}$ [1] which is a key feature in the attention mechanism. Thanks

[1]Here *s* is the length of the sample and *h* is the number of heads

to this mask we can let the attention layer prevent attention to some features and focus on others. In this respect, a data analysis has shown that most of the roles belongs to a small group of syntactic dependencies and postags with respect to the predicate. Therefore the attention mask has been built according to these two groups.

3 Model Architecture

3.1 Base Model

LSTMs are the network of the family of RNNs that better operate in sequential data, capturing pretty long range dependencies and are unbias toward recent inputs. Given a sequence of input vectors $\{x_n\}$ a bidirectional LSTM processes the sequence in opposite directions, and then combines the outputs:

$$\overrightarrow{\boldsymbol{h}}_t = \mathbf{LSTM}(\mathbf{x}_t, \overrightarrow{\boldsymbol{h}}_{t-1})$$

$$\overleftarrow{\boldsymbol{h}}_t = \mathbf{LSTM}(\mathbf{x}_t, \overleftarrow{\boldsymbol{h}}_{t-1})$$

$$\boldsymbol{y_t} = \overrightarrow{\boldsymbol{h}}_t + \overleftarrow{\boldsymbol{h}}_t$$

The input vectors are concatenation of 1-hot feature embeddings plus GloVe/BERT embeddings or a combination of the two. Finally a linear classifier does a linear transformation from the hidden dimension of the BiLSTM to the target space of the roles.

3.2 Attention Model

Thanks to the recent encoder-decoder transformer architecture, proposed by (Vaswani et al., 2017), in the paper ”Attention is all you need”, the concept of attention has increased its importance. Self attention, that is a special case of attention, has been applied successfully in many tasks, such as sentiment classification and textual entailment (Lin et al., 2017), machine translation and language understanding (Cheng et al., 2016) (Devlin et al., 2019), and many other tasks. Given the hidden dimension n and a matrix of n query vectors $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$, keys $\boldsymbol{K} \in \mathbb{R}^{n \times d}$, and values $\boldsymbol{Q} \in \mathbb{V}^{n \times d}$, the attention score is computed as follows:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V}$$

The matrices are projected in a linear transformation and then, h parallel heads, focus on different part of the value representation. A scaled dot product attention is used to compute the relevance

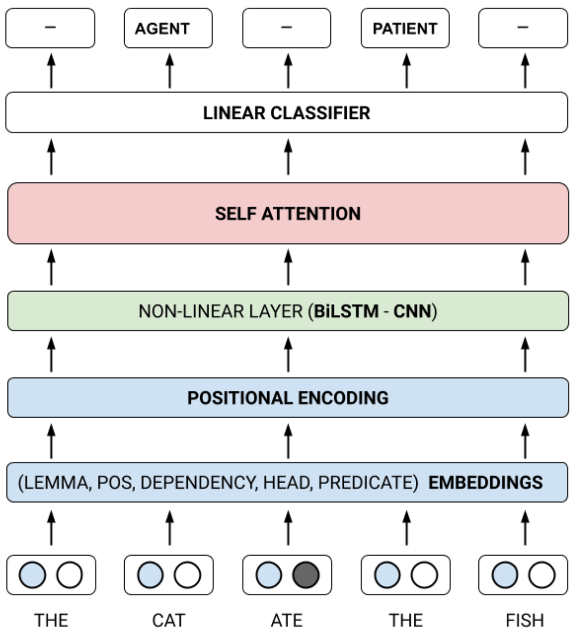
–
AGENT
–
PATIENT
–
LINEAR CLASSIFIER
SELF ATTENTION
NON-LINEAR LAYER (BiLSTM - CNN)
POSITIONAL ENCODING
(LEMMA, POS, DEPENDENCY, HEAD, PREDICATE) EMBEDDINGS
THE
CAT
ATE
THE
FISH

Figure 1: Attention model architecture.

between queries and keys. The results of the h parallel heads are finally concatenated and mapped to a final linear transformation:

$$\mathbf{M} = \text{concat}(\mathbf{M_1}, \dots, \mathbf{M_h})$$

$$\mathbf{Y} = \mathbf{MW}$$

We consider the self attention case because the matrices correspond each to the input sequence vector representation.

3.2.1 Non-linear Layer

Used as the only network layer, self attention does not achieves high performance. This is due to the fact that it only performs matrix multiplication through many linear transformations. It has been already found that adding non linearity to the model let the network understand and learn better the input features. Therefore, following (Tan et al., 2017) choices, I implemented two different non-linear layer placed in between the embedding and the self-attention layers.

Bi-LSTM As already seen in the base model, a bidirectional LSTM processes the input embeddings in two opposite directions, giving a non-linear representation of the input.

Convolution Convolutional neural networks (CNNs) have been demonstrated to compete with RNNs in many tasks, such as language modeling

(Dauphin et al., 2017), machine translation and speech recognition (Gehring et al., 2017) and also SRL (Tan et al., 2017). A first big advantage of CNN with respect to RNNs is an incredible speed up of the process, in fact, while the latter have to process sequences sequentially, CNNs do it simultaneously. The convolutional layer performs the following operation:

$$\text{Conv}(\mathbf{I}) = \sum_{0}^{C} \mathbf{W} \star \mathbf{I}$$

where $\star$ is the valid cross-correlation operator, applied between the weight matrix $\mathbf{W}$ and the input sequence $\mathbf{I}$, and C is the number of input channels (the embedding dimension). The *kernel* plays an important role in the convolutional layer, because it convolves k neighbouring words together.

3.2.2 Positional Encoding

The self attention mechanism does not distinguish between different positions. There are many ways to encode positions, like a simple positional embedding $\mathbf{p}$ with $p_i \in (0, 1)$. In this model, I tried another efficient method, proposed by (Vaswani et al., 2017) which is formulated as:

$$\text{timing}(t, 2i) = sin(t/10000^{2i/d})$$

$$\text{timing}(t, 2i+1) = cos(t/10000^{2i/d})$$

The timing signals are simply added to the input embeddings so that the input embeddings do not require an additional concatenation.

A complete overview of the attention model can be seen in figure 1.

3.3 Extra

The extra model performs argument identification and classification as well as predicate disambiguation. For this reason, the 1-hot embeddings are reduced, and the predicate encodings become a single flag $f \in \{0, 1\}$ indicating whether the current word is a predicate. It is very similar to the base model, less than than output classifiers, that are one for each of the two tasks. The model takes the GloVe/BERT embeddings (or a combination of the two) contatenated with 1-hot embeddings as input. Then a Bi-LSTM processes the sequences and the output feature is learned by the two linear classifiers. I tried both the base and the attention model, but the base one resulted in higher performance. The combination GloVe + 1-hot embeddings achieves the highest F_1 score. The performances are shown in the table 2

4 Experiments

In the following I describe the models settings and the results obtained with the various experiments.

4.1 Settings

Embeddings GloVe word embeddings have size 100, while two contextualized BERT embeddings have been tried, *bert-base-cased* with 768, and *bert-large-cased* with 1024 hidden sizes. [2]. The one hot encodings together have a total dimension of 554. For the base model, the best combination found is GloVe (100) + BERT (768) + 1-hot Embeddings (554), for a total of 1422 size, while for the attention model, the best (both for attention+LSTM and attention+Conv) are Glove (100) + 1-hot Embeddings (554) with positional encoding. The latter, compared to traditional position embedding combined with concatenation, has surprisingly achieved 2 more points on F_1 score. Also the attention mask played an important role, in fact, it increased the score of 3 points on F_1.

Hyperparameters The base model uses just a Bi-LSTM to learn the model parameters. Three hidden dimension have been tried, 256, 512, 1024, and the middle one achieved the highest F_1 score with 1 point difference compared to the other two. A dropout mask of 0.2 was applied, after the Bi-LSTM. This has encouraged the model to prevent from overfitting. For the attention model, the same choice was done for the Bi-LSTM layer, while the convolutional one required more tuning, given the considerable number of implementation possibilities:

- *Kernel size*. It is the filter applied to the input features convolved with the weight parameters. When applied to batches of data of encoded sequences, it establishes how many close features (neighbouring words) are captured for each convolution step. The size 3,5,7,9 where tried, which showed that $k = 5$ outperforms the others.

[2]The pretrained transformers models can be found on `https://huggingface.co/transformers/pretrained_models.html`

Model	**Variant**	Identification			Classification		
		P	**R**	**F1**	**P**	**R**	**F1**
Base	GloVe	0.954	0.954	0.954	0.913	0.915	0.914
h=512, b=768	BERT	0.964	0.945	0.954	0.921	0.904	0.913
	GloVe+BERT	**0.960**	**0.955**	**0.957**	**0.921**	**0.916**	**0.919**
Attention+BiLSTM	**GloVe**	**0.941**	**0.929**	**0.935**	**0.917**	**0.898**	**0.907**
h=512, pos=t, b=768	BERT	0.921	0.903	0.912	0.871	0.859	0.86
Attention+Conv	**GloVe**	**0.932**	**0.899**	**0.915**	**0.877**	**0.847**	**0.862**
k=5, p=avg, a=ReLU, pos=t	BERT	0.910	0.883	0.896	0.866	0.829	0.847

Table 1: Precision, recall, F_1 score of models with different variants, for both the tasks of identification an classification.h, b, pos, k, p and a stand for LSTM hidden dimension, BERT hidden dimension, positional encoding, kernel size, pooling operation and activation function, respectively.

- *Activation function*. Since convolution performs only linear transformations of the input, a non linear activation function is needed. Two different ones have been tried, the rectified linear unit (ReLU) and the gated linear unit (GLU) proposed by (Dauphin et al., 2017), which introduces a gate, like LSTM, to determine which part of the convolution to drop. ReLU worked slightly better, obtaining 1 point difference on F_1 score.

- *Pooling layer*. This layer is placed after the activation function of convolved input, and its purpose is to apply a pooling operation, as well as a kernel, and resulting in an halfed representation of the input space. Two operations have been tried, *max pooling* and *average pooling*. The latter outperforms the first one, slighlty improving the overall model performances.

Training Training is performed in parallel, thanks to the GPU power of Colab. To do so, the training batches have been padded to the maximum length of the corresponding dataset (143 for training dataset). The loss function used is *Cross Entropy Loss*, that combines log softmax and negative log likelihood. The optimization algorithm is also an important choice for the models. Adam (Kingma and Ba, 2017) optimizer combines the benefits of RMSprop and SGD with momentum, resulting in a faster and ensured convergence. Regarding learning rate, 0.005, 0.003, 0.001, 0.0005, have been tried in all the model variants, showing that 0.003 reached fast convergence and slightly better performances. There is no fixed number of epochs for the training phase, since it was performed with early stopping, until the validation set loss decreased, obtaining the best performance possible and preventing the model to overfitting.

4.2 Results

Table 1 shows the best hyperparameters combinations of the models, comparing precision (P), recall (R), and F1 scores obtained on the test dataset provided. Figures 2 and 3, show the confusion matrices relative to highest score variants of the two models, on the classification task.

5 Conclusion

This paper presents the two models for SRL task, with their features and results, in which I tried to demonstrate that not only LSTMs are good models for process word sequences. I am satisfied on the results all things considered, and I am sure that further improvements and implementation of the attention model can lead to a greater performance. One interesting approach might be a deep attention model, with residuals connections proposed by (He et al., 2015), also adopted by (Tan et al., 2017), that achieves high performances in the SRL task. Another interesting approach would have been graph convolutional network (GNC), that enhances the dependencies between predicates and roles. Although more effort was put in the attention model, LSTM based neural networks, even though slower, still obtain the highest performances on NLP sequence based tasks, such as SRL.

References

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. *Long Short-Term Memory-Networks for Machine Reading*. School of Informatics, University of Edinburgh, Edinburgh, Scotland.

Model	**Variant**	Identification			Classification			Disambiguation		
		P	R	F1	P	R	F1	P	R	F1
Multiprediction	**GloVe**	**0.968**	**0.937**	**0.952**	**0.906**	**0.877**	**0.891**	**0.912**	**0.912**	**0.912**
h=512, b=768	BERT	0.959	0.948	0.954	0.856	0.846	0.851	0.818	0.818	0.818
	GloVe+BERT	0.968	0.958	0.963	0.878	0.863	0.870	0.867	0.847	0.847

Table 2: Precision, recall, F_1 score of the multiprediction model with different variants, for the tasks of argument identification and classification and predicate disambuguation.h and b stand for LSTM hidden dimension and BERT hidden dimension respectively.

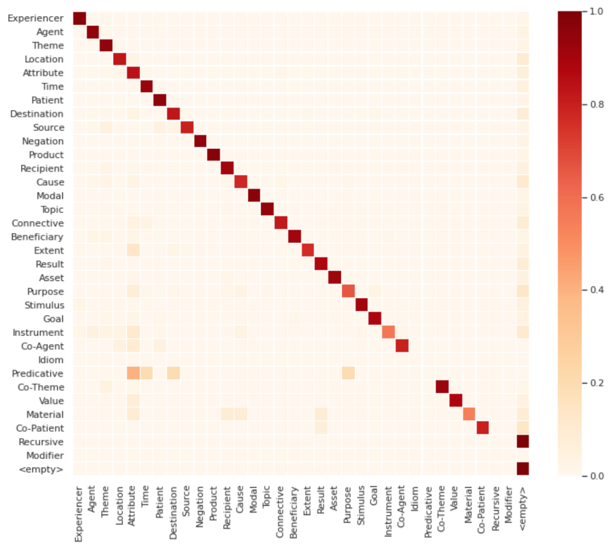

Figure 2: Confusion matrix for argument classification of simple model.

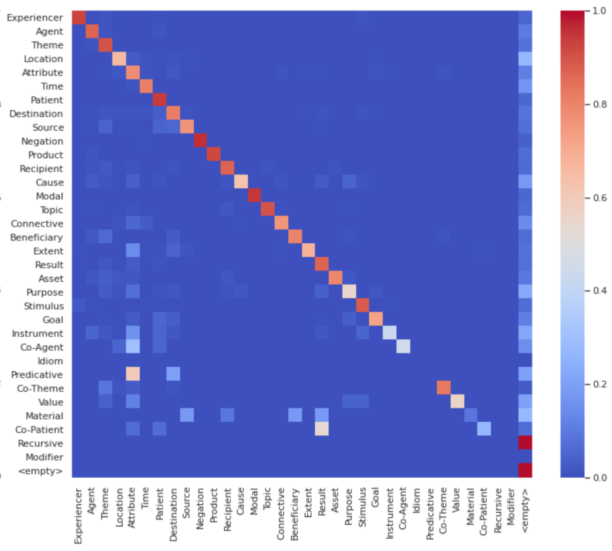

Figure 3: Confusion matrix for argument classification of attention model.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. *Language Modeling with Gated Convolutional Networks*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Google AI Languagel.

Andrea Di Fabio, Simone Conia, and Roberto Navigli. 2019. *VerbAtlas: a Novel Large-Scale Verbal Semantic Resource and Its Application to Semantic Role Labeling*. Sapienza University of Rome, Italy, Rome, Italy.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. *Convolutional Sequence to Sequence Learning*. Facebook AI Research.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. *Deep Residual Learning for Image Recognition*. Microsoft Research.

Diederik P. Kingma and Jimmy Lei Ba. 2017. *Adam: a method for stochastic optimization*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. *A Structured Self-Attentive sentence Embedding*. Montreal Institute for Learning Algorithms (MILA), Montreal, Canada.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. *GloVe: Global Vectors for Word Representation*. Stanford University, Stanford, California.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2017. *Deep Semantic Role Labeling with Self-Attention*. School of Information Science and Engineering, Xiamen University, Xiamen, China.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. *Attention Is All You Need*. Google Research.