MNLP Homework-1
Event Detection

Francesco Palandra
Department of Computer Science
Sapienza University of Rome
palandra.1849712@studenti.uniroma1.it

Abstract

In this report, it will be presented a comparison of various models for an event detection task, focusing in depth on the effect of different hyperparameters. It has been utilized a BiLSTM architecture with a CRF for the final classification and character-level informations are incorporated through a CNN for character embeddings. Lastly a pretrained POS detector is used to enhance word representation.

1 Introduction

Event Detection is a common NLP task that involves identifying specific occurrences or patterns within text data. With the advent of transformers (Vaswani et al., 2017), BERT has become a popular choice for high performance in many NLP tasks, for the model choice BiLSTM are been widely used due to their ability to capture temporal information. For the final classifier layer sequence labeling models, such as Hidden Markov Models and Conditional Random Fields (CRF) seems to show better results rather than a softmax. The hyperparameters have a significant role as a good configuration can have a relevant impact on the model's performance (Reimers and Gurevych, 2017), differentiating between mediocre and state-of-the-art results.

2 Embeddings

Initially, in the data preprocessing stage, words were converted to lower case, stopwords were eliminated, and a lemmatizer was utilized to ensure that similar words were not treated as separate entities. However, the stopwords removal and the use of a lemmatizer resulted in worse performance, so it has been decided to not use them anymore. For the embeddings three distinct pre-trained models from gensim were tested: word2vec, glove and fasttext, each one having an embeddings size of 300.

3 Model Architecture

The proposed model architecture incorporates various techniques to improve accuracy. In terms of the model input, a batch of sentences is taken and padded to have sentences of the same length. These processed sentences are then fed into a BiLSTM layer, which generates an output that passes through a classifier (softmax) that maps the output onto the 11 possible classes. During the forward pass, a Cross Entropy Loss is computed to estimate the error rate of the prediction. The loss is then backpropagated through the network and the optimizer updates the model's gradient. From this architecture then, many features can be added in order to improve the overall performance.

3.1 CRF

Several studies have demonstrated that adding a CRF as a classifier in the last layer of a neural network can significantly improve its performance (Ma and Hovy, 2016), due to the creation of dependencies between nearby words, which is sensible in a linguistic context. In the implementation of the BiLSTM-CRF model, a simpler variant known as Linear CRF is utilized, where the dependency between labels is modeled as a linear chain. CRF returns the negative log likelihood:

$$P(y|x) = -\frac{1}{Z(x)} \cdot exp(\sum_{i=1}^{l} U(x_k, y_k) + \sum_{i=1}^{l-1} T(y_k, y_{k+1})) \quad (1)$$

where $U(x_k, y_k)$ represents the emission, so the score of label y given the embedding x at the k-th timestep, $T(y_k, y_{k+1}$ is called transition score and creates a linear dependency between nearby labels and $Z(x)$ is a normalization factor (the one that we have also in the softmax).
Lastly, CRF uses the `viterbi_decode` in order to find the most likely sequence,that is equivalent to find the sequence that maximizes the log likelihood p(y|x).

Parameter	Value	Parameter	Value
embeddings	glove	optimizer	Adam
classifier	crf	lr	1e-4
# LSTM	2	dropout	0.4
hidden size	1024	clip	1
batch size	64	unfreeze	10

Table 1: List with the best hyperparameters found

3.2 BiLSTM-CNN-CRF

Furthermore, it has been observed that adding information at the character level can help to improve the representation of words as embeddings (Chiu and Nichols, 2016), especially for words that are not present in the dictionary. To achieve this, an embedding of arbitrary length (usually between 30-70) is randomly initialized for each character. Within the model, a vector of embeddings obtained through a Convolutional Neural Network of the word's characters is concatenated to the embeddings of each word and then passed to a BiLSTM layer. To further enrich the embeddings, a pretrained Part of Speech (POS) model is used to determine the role of each word in the sentence. So, an additional vector in the form of one-hot encoding of POS tags is concatenated with the embeddings, allowing the model to better understand the role of each word and improve its performance in the prediction.

4 Evaluation

The evaluation stage follows the training phase, during which the model is tested on a validation set to evaluate its performance. In this stage, the f1-macro metric is used as evaluation metric due to its effectiveness in measuring the model's overall performance. The evaluation process also includes the analysis of hyperparameters, which are tuned to optimize the performance of model architecture.

4.1 Hyperparameters Tuning

For the hyperparameter tuning, initially, a random search was conducted to analyze the general behavior and performance of the model. This involved running multiple trials with randomly selected parameters from a pool. Once a general understanding was gained, each hyperparameter was analyzed individually to determine its impact on the model's performance. In addition to hyperparameter tuning, various optimization techniques, including gradient normalization, adaptive learning rate and fine-tuning of the embeddings were used to improve the model's performance.

4.2 Softmax vs CRF

As previously mentioned in Chapter 3.1, the use of CRF as the final classifier results in a potential improvement in performance, as shown in the violin plot (1). The plot displays the estimated F1-score value on the y-axis as a probability density function derived from the data, thereby describing the contents of the violin plot. The results show a slight improvement when using CRF, with the mean of the pdf being slightly higher and the standard deviation being lower compared to the 'softmax' case, indicating superior performance.

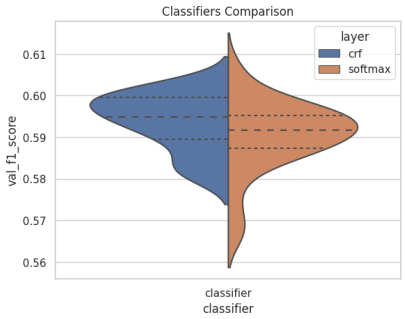

Figure 1: Violin plot that show the different performance of the classifiers,

4.3 Optimizer

Several optimizations have been evaluated for the neural network, including Stochastic Gradient Descent (SGD), Adam, and Nadam. The results show (Figure 2) that Nadam faced difficulties in converging to a minimum in many cases, while the other optimizers performed better using their recommended settings. Specifically, Adam and SGD provided more stable and superior results than Nadam.

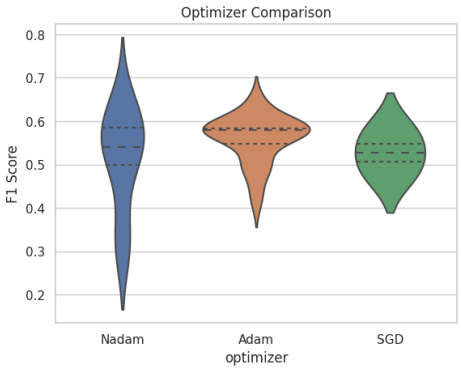

Figure 2: Violin plot showing the comparison of different optimizers with different lr.

5 Results

The results of the three models depicted in Figure 4 demonstrate the f1-score and validation loss. The observed difference between the models is minimal, however, the incorporation of the CRF enables the network to learn more rapidly and converge to a 0.59 f1-score in validation, while simultaneously reducing the loss to 0.13. Similarly, the inclusion of Char and POS features facilitates the model in achieving an improved f1-score of 0.60 with a corresponding validation loss of 0.12. However, the observed improvements may be small due to the limited number of "I" events, as illustrated in Figure 3, which may impede the performance of the CRF. The confusion matrix (Figure 7) presents the results of the BiLSTM-CNN-CRF + POS model. As expected, the "O" label was accurately detected, given its high frequency and clear differentiation from events. The same holds for "B" events. However, the performance for "I" labels varied depending on the specific event type. In particular, two event types posed challenges to the network, likely due to their relatively low representation in the dataset. Notably, the "B" class achieved satisfactory results in these cases.

6 Conclusion

To sum up, it has been explored various techniques and model architectures for event detection in natural language processing. The results suggest that the incorporation of CRF, character-level information through CNNs, and pre-trained POS embeddings can lead to improved performance in certain cases. While the improvements were not substantial, they still indicate the potential benefits of these approaches.

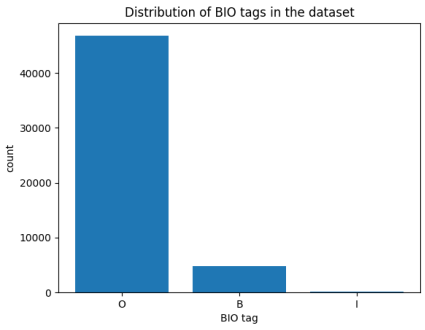

Figure 3: Distribution of the BIO label in the dataset shows that the "O" class is dominant, which is consistent with a natural language. In contrast, while other classes such as "I-POSSESSION" are underrepresented.

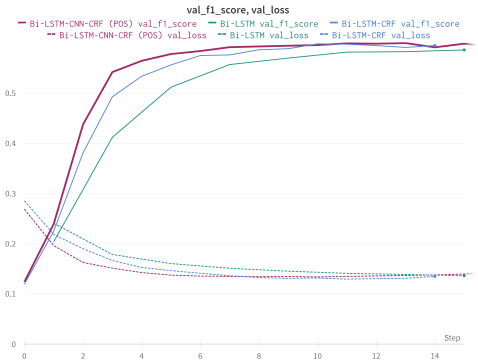

Figure 4: f1-score and loss in validation for the three models

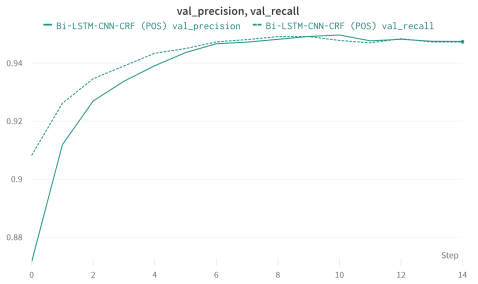

Figure 5: Precision and Recall on the final Model

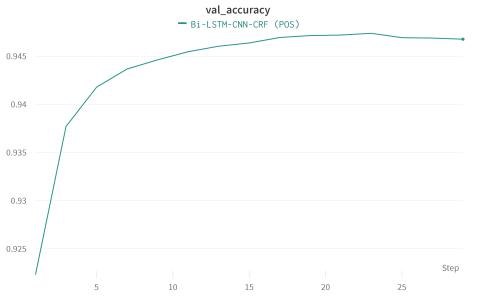

Figure 6: Accuracy of the final model

References

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf.

Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks.

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

Figure 7: Confusion Matrix that plots in the x-axis the prediction and in the y-axis the true value. The value are normalized such that the some in a line gives 1.

Model	Optimizer		LSTM Units			Dropout		Clip			
	Adam	SGD	512	1024	2048	0.1	0.4	1	5	**+POS**	**f1**
BiLSTM	0.57	0.49	0.55	0.57	0.57	0.55	0.56	0.56	0.56	0.57	0.57
BiLSTM-CRF	0.58	0.48	0.56	**0.59**	0.58	0.56	0.57	0.57	0.56	0.58	0.59
BiLSTM-CNN-CRF	**0.58**	0.50	0.57	0.59	0.59	0.58	**0.59**	**0.59**	0.58	**0.60**	**0.60**

Table 2: Table with the list of some hyperparameters and their results. Just the parameter selected was changed so the number in table corresponds on the effect of the single parameter.

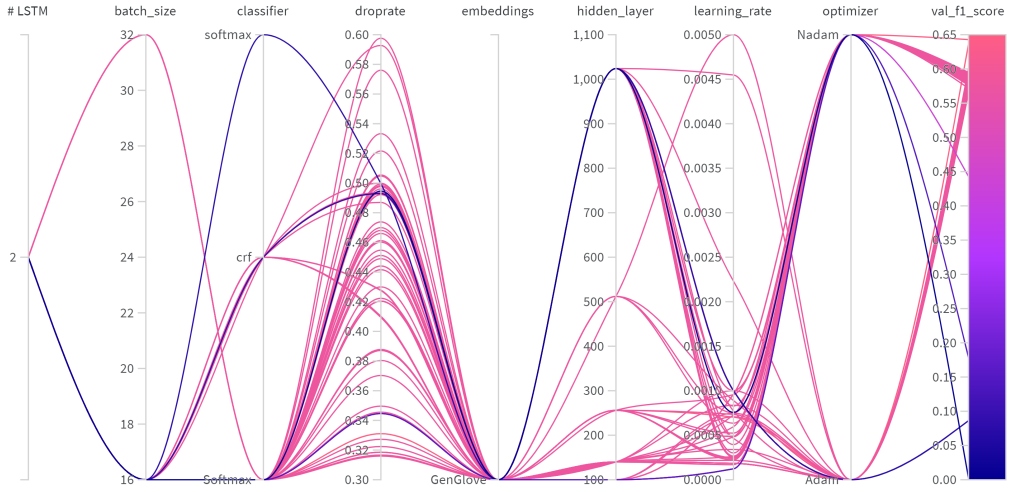

Figure 8: This image shows the the gait of the random search with different parameters. The softmax generally results in worse perfomance.

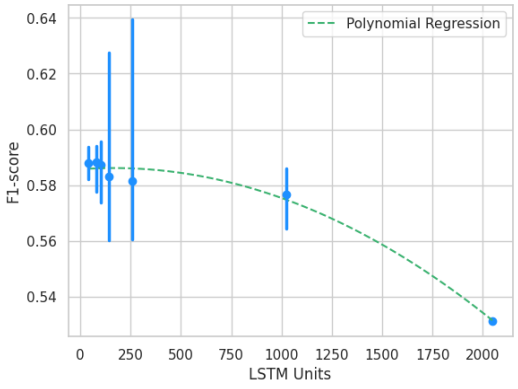

Figure 9: In this plot different lstm units were tried in the BiLSTM classic model and a polynomial fit is presented. However due to some overfit in the case of low number of lstm units the plot suggest that the best decision was 128, while with a deep analysis it turned out it was 1024.

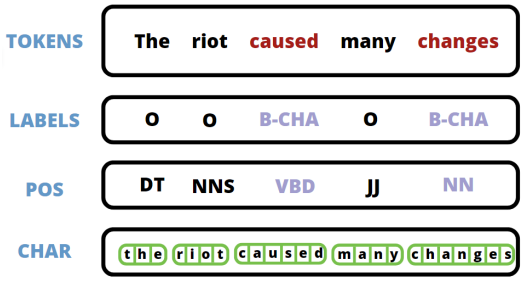

Figure 10: Input data for the model, where tokens,POS tags, and chars are concatenated to generate embeddings that are fed into the LSTM network. The labels associated with each input are utilized for computing the loss function.