Assignment 1

Alessio Cocchieri, Riccardo Grassi, Marco Guerra and **Alessio Bonacchi**
Master's Degree in Artificial Intelligence, University of Bologna
{ alessio.cocchieri2, riccardo.grassi5, marco.guerra20, alessio.bonacchi }@studio.unibo.it

Abstract

In traditional grammar, a part-of-speech (POS) is a category of words that have similar grammatical properties. The part of speech explains how a word is used in a sentence. The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, or simply POS-tagging. There are different techniques for POS Tagging, among which we can distinguish Lexical Based Methods, Rule-Based Methods, Probabilistic Methods and Deep Learning Methods. In this paper we will discuss about the implementation of a Recurrent Neural Network (RNN) aimed at predicting the POS tag for each word, given a corpus of documents, by using GloVe's word embedding. We started by defing a simple baseline model and then we tested it along three different variations.

1 Introduction

In natural language, to understand the meaning of any sentence we need to understand the proper structure of the sentence and the relationship between the words available in the given sentence. To overcome this issue, we need to learn POS Tagging. POS tagging is a supervised learning solution which aims to assign parts of speech tag to each word of a given text (such as nouns, pronoun, verbs, adjectives, and others) based on its context and definition. Part-of-Speech tagging in itself may not be the solution to any particular NLP problem. It is however something that is done as a pre-requisite to simplify a lot of different problems. Indeed, it finds applications, for instance, in Named Entity Recognition (NER), sentiment analysis, question answering, and word sense disambiguation. In this work we propose a starting baseline model which consists of a Bidirectional Long Short-Term Memory Recurrent Neural Network (BiLSTM-RNN) with a Dense/Fully-Connected layer on top. It is well known that BiLSTM-RNNs are very effective for tagging sequential data e.g. handwritten documents. Successively, we considered three variation of the baseline model: replacing LSTM layer with a GRU layer, adding an additional LSTM layer, adding an additional dense layer. Eventually, we evaluated the performances on the test set only for the best two models (according to the results obtained on the validation set). The metric we adopted to evaluate the final model is the F1-macro. It is indeed a good indicator for unbalanced datasets as the one we are considering in our task. During the training process however we used accuracy as indicator since there is no way to aggregate "partial" F1 scores computed on mini-batches.

2 System description

2.1 Words embedding & OOV

Word Embeddings encode the relationships between words through vector representations of the words such that words that are closer in the vector space are expected to be similar in meaning. For the representation of the words we adopted the pre-trained GloVe embeddings. We decided to use vectors of size 300 as embedding vectors to get the best possible representation of the words. To handle the embedding of the Out-Of-Vocabulary words (words not in GloVe vocabulary), instead, we decided to represent each of them with vectors of 300 random values in the interval $[-0.5, 0.5]$.

2.2 Baseline model

The starting point to solve our POS tagging task has been the baseline architecture. The model consists of an initial non-trainable Embedding layer which also takes into account the masking of the 0s used for padding, followed by a BiLSTM layer of 128 units and a final dense layer of 46 units. We chose to dropout the 50% of nodes of each LSTM layer.

2.3 Basemodel variations

1. **Replacing of LSTM with GRU layer (BiGRU):** A bidirectional GRU layer has been replaced to the previous bidirectional BiLSTM with the same number of hidden nodes. GRU is a simpler model than LSTM, easier to train and therefore uses less memory and executes faster than LSTM. Here we still decided to drop out 50% of the nodes of the GRU layer.

2. **Additional LSTM layer (2BiLSTM):** We added a further BiLSTM layer to the baseline model composed by 128 hidden nodes. Moreover, we opted for dropping out 50% of the nodes of the first BiLSTM and for dropping out the 25% of the nodes of the second BiLSTM layer.

3. **Additional dense layer (BiLSTM+D):** We added an additional dense layer to the base line model and decided to use *tanh* as activation function since it performs well in RNN. The number of hidden node was settled to 128.

3 Experimental setup and results

We experimented the baseline model and all its variations with different hyperparameters and optimizers. Table 1 summarizes the final configuration for all the different models (H1 = first hidden layer units, H2 = second hidden layer units, D1 = dropout applied to H1, D2 = dropout applied to H2).

Model	H1	D1	H2	D2	Batch
BiLSTM	128	0.5	-	-	512
BiGRU	128	0.5	-	-	512
2BiLSTM	128	0.5	128	0.25	512
BiLSTM+D	128	0.5	128	-	512

Table 1: Models tuning

In our case, the maximum length of the sentences may be considered as a further hyperparameter. The majority of the sentences had a length lower than 60, therefore we decided to use 60 as limit to truncate them. Such length can also be used for the evaluation phase, since the maximum length of test set sentences is always shorter. In a real-case scenario, where the test set is unknown, we should be careful when managing the test set to make inference, since we cannot truncate its sentences, otherwise it would change it, altering the performances.

For all the experiments, as loss, we used *categorical cross-entropy*, since we worked with one-hot encoded labels, with *softmax* activation function for the last layer. As optimizer we tested both Adam and RMSprop with an initial learning rate equal to 0.0015. Adam performances turned out the best, therefore we eventually decided to use the latter.

The callbacks considered for our experiments have been Early Stopping with patience 10 and Reduce On Plateau. The latter allowed to halved the learning rate during the training each time the loss on validation set does not increase for 3 consecutive epochs. Finally, the best models turned out to be the baseline and 2BiLSTM. In table 2 it is possible to see the performances of the two models in both validation and test sets.

Model	F1 val	F1 test	Epochs
Baseline	0.72	0.79	54
2BiLSTM	0.71	0.79	74

Table 2: Best models evaluation

4 Discussion

The final F1-scores have been calculated by getting rid of the punctuation/symbol classes and of the classes with null support in the test set.

The best performances we get on test set compared to validation set may be justified by:

- train data are more similar to test data instead of to validation data;
- During evaluation, the test set considers fewer classes than validation set, getting rid of those classes with low F1 score in validation set;

5 Conclusion

The results obtained have shown that both Baseline and 2BiLSTM allowed us to achieve the best performances having actually the same results on both validation and test set.

As future developments we might experiment:

- deeper models to enhance the representation capacity;
- a larger number of training data to help reducing the overfitting and enhancing the final results;
- other optimizers with different settings (e.g. different initial learning rate)