Assignment 1 - POS tagging

GIUSEPPE BOEZIO
Università di Bologna
giuseppe.boezio@studio.unibo.it

SIMONE MONTALI
Università di Bologna
simone.montali@studio.unibo.it

GIUSEPPE MURRO
Università di Bologna
giuseppe.murro@studio.unibo.it

Abstract

We compared the performance of POS tagging models using different types of networks and parameters, namely BiLSTM, BiGRU, double BiLSTM with different combinations of fully connected layers. The input was embedded using GloVE embeddings [1], and the Out-Of-Vocabulary words were randomly instantiated. After sufficient hyperparameter tuning, the best models were BiLSTM and BiGRU, and they achieved Macro-F1 scores of 0.6183 and 0.6117.

1 Introduction

In this assignment, we will implement a POS tagging model using a Bidirectional Long Short-Term Memory neural network. The model is trained on the Dependency-parsed Penn Treebank corpus [2], which though contains a full dependency tree and not just the tags.

2 POS tagging

This paper will present the usage of bidirectional models, able to better learn the positional information of sentences, proceeding forward and backward along the sentence. It's interesting to show a distribution of the most present tags, information which may be particularly useful in the error analysis of the model.

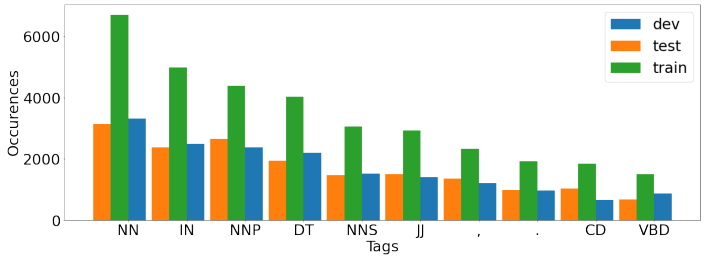

Figure 1: POS tag distribution

3 Data input and preprocessing

The first task to perform in every machine learning task is represented by the loading of data and its preprocessing. Three classes were created to tackle this problem: `DataInput`, `TextVectorizer` and `TargetVectorizer`.

3.1 DataInput

DataInput is responsible for the mapping between raw text data and Python data structures. It first downloads the dataset and then splits the data in three sets, namely the training set, validation and test set. The first will be used for training, the second to evaluate the different models and their parameters, and the latter will be used to test the picked models in the end. This class also splits the documents into sentences (which proved itself to have better performance then just keeping the documents) and converts the words to lowercase.

3.2 TextVectorizer and TargetVectorizer

We cannot just feed the neural networks with text. While sparse representations were used a lot in the past of NLP, the current state of the art requires us to use dense embeddings like GloVE [1]. Our `TextVectorizer` class, therefore, downloads these embeddings, parses them using a dictionary and computes the numerical representations for the tokens. Several embedding dimensions are available, and the 100-dimensional embedding proved itself to be a good tradeoff between complexity and informativeness. `TargetVectorizer`, instead, computes a one-hot encoding of the target classes, and provides a simple method to perform the inverse transform too. Note that both the methods apply zero-padding up to a maximum length, which will then be masked with a `keras.layers.Masking` layer in the neural networks.

3.2.1 OOV words and holdout

OOV words need some special attention: computing their embedding in the wrong way could be a source of data leakage, something that we want to avoid as much as possible in machine learning. For this reason, we implemented a `TextVectorizer.adapt` method that, given a dataset, computes the out-of-vocabulary words and their embedding (which is just represented by a random array). This will then be sequentially used on the training, validation and test sets, keeping these steps separate to simulate a real-world task, in which the *test set* represents sentences that we never heard at training time. These sets, formally representing the *holdout* technique, will allow us to understand the performances of our model on data that was never seen by the model. The validation set is provided to the neural network training to have a measure of how general the model is, and is particularly useful to prevent overfitting through early stopping.

4 Models

Several recurrent models are seen in the literature for NLP tasks, and we picked two different types of recurrent layers (bidirectional), namely Long Short-Term Memory [3] and Gated Recurrent Units [4], together with one or two Dense layers. Summing up, 4 models were tried:

- Bidirectional LSTM + Dense layer
- Bidirectional GRU + Dense layer
- Two bidirectional LSTM + Dense layer

- Bidirectional LSTM + Two Dense layers

Each of these models contained a first masking layer too. The models have then been compiled with a categorical cross-entropy loss, representing the difference between the real probability distribution of the class (the one-hot encoding) and the prediction of the model. The last layer allows us to output a probability distribution for the classes, using a softmax activation. Note that because of this activation, the padding will have a uniform probability distribution over all classes, and we can filter that basing on the standard deviation of the probabilities (which is close to 0 for padding). All these layers have been tuned using kerastuner, obtaining the optimal configurations through HyperBand [5]. We can take a look at the history of the trainings to understand the differences in performances. We can notice a slight overfitting 2 in all the methods, which could probably be solved with a larger dataset, as 100 documents for the training set are probably not enough. All of the networks use an *Adam* optimizer [6], a second order optimization method that exploits momentum techniques and per-parameter learning rates (particularly useful for NLP problems, in which we may have sparse gradients).

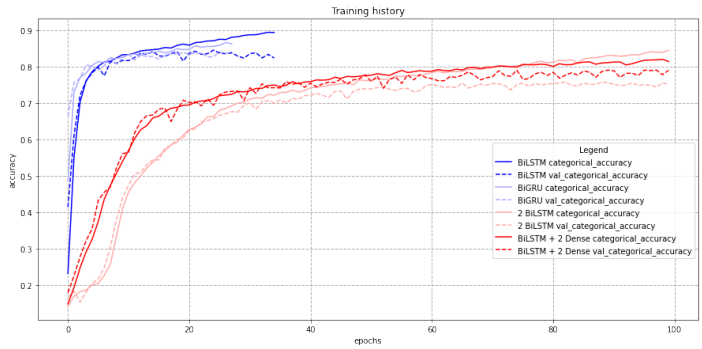

Figure 2: Accuracy during the training

5 Performances and metrics

5.1 Metrics

The metric which has been used to compare models is the macro F1 score with the validation set, results are the following:

Metric	BiLSTM	BiGRU	2 BiLSTM + Dense	BiLSTM + 2 Dense
Accuracy val	0.8457	0.8362	0.7564	0.7907
F1 score val	0.6183	0.6117	0.5094	0.4840
F1 score test	0.6246	0.6062	not tested	not tested

As shown in the table above, the best models are BiLSTM and BiGRU, therefore the error analysis has been performed on these two models. The two other models perform worse, maybe because too many dense layers are not useful, as they do not take into account time, whereas two BiLSTM could be too complex and cause overfitting.

5.2 Error analysis

5.2.1 BiLSTM

POS tag	precision	recall	f1-score	support
RBS	0.75	0.50	0.60	6.0
VBG	0.58	0.62	0.60	321.0
JJR	0.59	0.50	0.54	108.0
RP	0.44	0.51	0.48	37.0
RBR	0.40	0.42	0.41	19.0
JJS	0.85	0.25	0.39	44.0
WRB	0.75	0.18	0.29	34.0
WP$	0.00	0.00	0.00	5.0
PDT	0.00	0.00	0.00	9.0
NNPS	0.00	0.00	0.00	83.0

As we can see from the table above, there is not a correlation between the support of the labels and the values of the metrics. Infact, for example, the support of possessive pronouns (PRP$) is very small compared to the nouns' (NN) one but the value metrics are quite higher in the first case.
Interesting errors are represented by the predeterminers (PDT) which are in most of the cases missclassified as prepositions or subordinating conjunctions. This could seem reasonable because they are commonly used before nouns therefore could be misclassified as prepositions.
It is also important to highlight the class of proper plural nouns (NNPS). Looking at the confusion matrix it is possible to notice how a lot of labels are classified as plural nouns or proper singular nouns. This could be explained by the fact that the lowercase folding is useful for some labels, but for others like this one could be a problem because this class is characterized by the fact that all words begin with an upper case letter, therefore in this particular situation the case folding is not recommended.
A particular situation is represented by the superlative of adjectives (JJS). In this case we notice high precision but a low recall, meaning that there are not a lot of labels which are misclassified as superlatives but a lot of superlatives are misclassified as adjectives. This could be explained by the fact that, in English, some adjectives have more than two syllables and therefore they are preceded by "most" which does not change the form of the adjective.

5.2.2 BiGRU

Results are very similar compared to the ones of the previous model. As in the previous case there is a problem with superlative but this time with adverbs (RBS). Most of them are classified as adjectives (JJ) and not as adverbs as it could be expected and this could be caused also by a low support. Also in this model there are problems to classify correctly adjective superlatives as shown by recall value. Another interesting error is represented by possessive wh-pronoun (whose, tag WP$) which is classified as adjective (JJ). This maybe happens because both labels are used in front of nouns to express something about it. NNPS and JJS suffer of the same problems described previously. This model presents the same problem in classifying predeterminers (PDT) and this time this label is sometimes misclassified as determiners (DT) which could be caused by a semantically similarity between these two labels.

References

[1] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[2] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993. [Online]. Available: https://aclanthology.org/J93-2004

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[4] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[5] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, p. 6765–6816, jan 2017.

[6] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980