Assignment 1
NLP Course Project

Daniel Bernardi, Daniele Santini, Hiari Pizzini Cavagna and **Muhammad Saleem Ghulam**
Master's Degree in Artificial Intelligence, University of Bologna
{ daniel.bernardi, daniele.santini2, hiari.pizzinicavagna, muhammad.ghulam }@studio.unibo.it

Abstract

Comparative analysis of Neural-Network based implementations of sequence labeling: combining Dense, LSTM and GRU architectures for Part-Of-Speech tagging.

1 Introduction

The goal of this assignment was to perform Part-Of-Speech tagging using neural architectures on the Dependency Treebanks corpus from NLTK.

The necessary tasks included the download, preprocessing, and analysis of the corpus, the dense embedding of the words in the corpora and the comparation of the results of four Neural Network architectures choosen ahead of time:

- Bidirectional LSTM layer + Dense layer (baseline)
- Bidirectional GRU layer + Dense layer
- 2x Bidirectional LSTM layer + 1x Dense layer
- 1x Bidirectional LSTM layer + 2x Dense layer

The intended evaluation metric on the test set is F1-Macro, without considering punctuation classes.

2 System description

The corpus download is implemented in plain Python. Since the corpus files are numbered and the train-val-test split is done before the data loading by selecting a range of files, we have implemented a function (called loadCorpus) which takes in input the range of files to open and loads these as a Pandas DataFrame.

The Exploratory Data Analysis and data preprocessing are implemented using Pandas and Matplotlib.

As per instructions, the word embedding was done using the GloVe dense embedding (Pennington et al., 2014). This was implemented using the glove-wiki-gigaword-200 embedding model [7] from the gensim library (Řehůřek and Sojka, 2010) and using it inside a Tokenizer layer that was used ase the first layer in our Tensorflow neural network. The Out-Of-Vocabulary words were handled by adding a random embedding vector to the embedding matrix.

Using Python, Numpy and Tensorflow interfaces we made sure to make our code reproducible, then we proceeded to implement the four neural network architectures with Tensorflow.

3 Data

The corpus contained training, validation and test data. As mentioned above, it was divided in 200 numbered files and the train-val-test split is done before the data loading by selecting a range of files.

Based on the information available on punctuation [7], we have identified these punctuation classes: ``, ", -LRB-, -RRB-, ,, ., :, HYPH, \#, \$, SYM, ''. The distribution of these classes among all classes can be observed in the Python notebook.

4 Experimental setup and results

We first experimented manually tuning the hyperparameters for these four models.

Then we calibrated the configurations for the four different architectures using Keras Tuner [7]. The hyper-parameters considered and their range (min, max) were: the number of units (16, 256), the activation functions (relu, tanh), dropout (0, 0.2) and learning rate (1e-4, 1e-2), with Adam as optimizer. The chosen tuner algorithm was Hyperband (Li et al., 2016), that allows to quickly converge on a high-performing model, comparing possible combinations through a "championship style" bracket.

Architecture	Units	Dropout	Activation	LR	Epochs	Acc.	F1 Val.
Bi-LSTM + Dense	32	0	tanh	28e-4	9	0.885	0.707
Bi-GRU + Dense	112	0.05	tanh	64e-5	17	0.891	0.701
Two Bi-LSTM + Dense	128+112	0+0.2	tanh+relu	14e-4	7	0.890	0.708
Bi-LSTM + Two Dense	112+64	0	relu+tanh	12e-4	6	0.885	0.716

Table 1: Results obtained using Keras Tuner for every different architecture

The algorithm used accuracy to determine the best model, training at most for 30 epochs, using the Early Stopping callback with a min. value of 15e-4 to increase the performances. The results obtained and the F1-score calculated are showed in the Table 1. The F1-score is macro-averaged, calculated excluding all the punctuation classes. In the end, the F1-score on the test set has been calculated, both best models had a better score compared with the validation score.

Model	F1 Val.	F1 Test
2xBi-LSTM + Dense	0.708	0.782
Bi-LSTM + 2xDense	0.716	0.793

5 Discussion

All the models ended up with a F1-score result on the test set around 0.8, good but not perfect.

Since Gated Recurrent Units are less complex than Long Short Term Memory units we weren't surprised to see that the GRU model didn't outperform the baseline LSTM model.

During the initial manual phase we started with a high number of units in the first Dense layer of the last model and noticed that it was particularly susceptible to over-fitting. This problem was addressed and solved by using dropout and early stopping. KerasTuner then demonstrated that a better result could be obtained simply reducing the number of units in the dense layer while keeping early stopping. The optimized hyper-parameters also changed the learning rate and the activation functions, which could have an impact.

As we expected the best performing architectures were the last two, however they didn't substantially outperform the baseline, even after tuning the hyper-parameters with Keras Tuner.

6 Conclusion

Since this problem relies heavily on identifying the relationship between the elements in the string, a possible way to improve the output would be to use attention-based architectures, such as transformer-derived techniques like BERT. Recurrent architectures like LSTM and GRU are partially able to learn these patterns but attention based architectures are precisely designed to exploit these relationships.

7 Links to external resources

- Corpus
- English punctuation on Wikipedia
- PUNCT POS tags on Universal Dependencies
- gensim library docs
- gensim available models
- KerasTuner API - Keras
- Keras Tuner - Tensorflow
- Our GitHub repository

References

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. Hyperband: A novel bandit-based approach to hyperparameter optimization.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.