POS Tagging Using Neural Architectures

Mohammad Amin Nazerzadeh, Davide Baldelli, Mohammad Reza Ghasemi Madani
Master's Degree in Artificial Intelligence, University of Bologna
{ davide.baldelli4, mohammad.nazerzadeh, mohammadreza.ghasemi}@studio.unibo.it

Abstract

Text corpora tagged with Part-of-Speech (POS) information are helpful in many areas. In this work, we tuned and evaluated a few modifications of a baseline BiDirectional LSTM architecture on a corpus annotated with POS tags. The best models (LSTM and GRU) performed roughly 0.83 and 0.56 considering the Accuracy and Macro F1-Score criteria respectively. We then analyzed the results and investigated possible causes for the errors.

1 Introduction

Many natural language tasks require the accurate assignment of POS tags to previously unseen text. Due to the availability of large corpora which have been manually annotated with POS information, many taggers use annotated text to learn either probability distributions or rules and use them to automatically assign POS tags to unseen text. One of the most extended approaches nowadays is the statistical family. Basically, it consists of building a statistical model of the language and using this model to disambiguate a word sequence. The language model is coded as a set of co-occurrence frequencies for different kinds of linguistic phenomena which is usually found in the form of n-gram collection. Although the statistical approach involves some kind of learning, supervised or unsupervised, of the model's parameters from a training corpus, we place in the machine learning family only those systems that include more sophisticated information than an n-gram model (Marquez et al., 2000). In this work, we present a machine-learning approach to learn tags on unseen data based on what we mentioned above.

2 System description

For the data preparation, we first need to do some pre-processing operations. Primarily, we transform document-level division to sentence-level division.

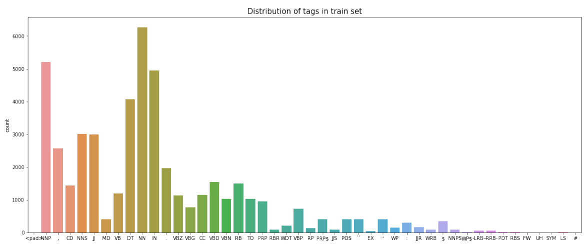

Figure 1: Distribution of tags in train set

The next step is to pad vectorized sequences to feed them to the network. There are two approaches that can be utilized for this task. First, we use the sequences' length distribution and choose a threshold for the maximum length such that it includes most of the data (%95 or higher) and then pad or truncate the sequences to the chosen length. Fig. 2 shows the sequences' length histogram over the train set. In our dataset, %98 of the sequences have a length of less than 50. We can also do more efficiently by applying variant length padding by the means of collate function that PyTorch provides. This method needs less computation and is faster than the former especially when bidirectional elements come along. Ultimately, we use GloVe embeddings and map sequences/tags tokens to word ids and their embedding vectors. In the following section, we will go through the models' architecture we used for the POS tagging task.

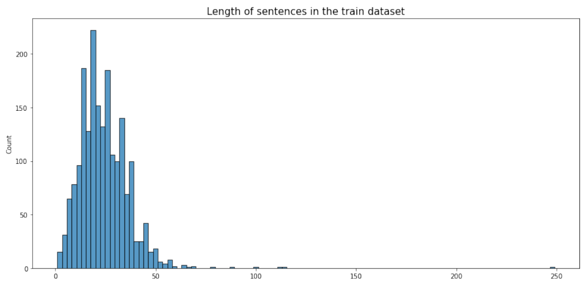

Figure 2: Sequence length histogram

In this work, we aim to implement four different models and compare their performance on POS tagging tasks. In the baseline model, we utilize an Embedding layer, an LSTM layer, and a Convolutional layer as a time-distributed dense (TDD) layer. What the TDD layer creates, is a fully-connected (dense) layer that is applied separately to every time step during GRU/LSTM Cell unrolling. So the error function will be between the predicted label sequence and the actual label sequence.

In the training, since we are dealing with a multi-class problem, cross-entropy loss has been taken into account as the loss function. We also utilize the macro f1-score and accuracy in the evaluation phase as metrics to have a better sense of the model behavior.

3 Experimental setup and results

In the previous section, we discussed the baseline model architecture. Here, we also experiment POS tagging using different models in which the architecture varies a bit. As an instance, in one of the models, we replaced LSTM layer with GRU. GRU was first introduced by (Cho et al., 2014). It is the newer generation of RNNs and is pretty similar to an LSTM. GRU has got rid of the cell state and used the hidden state to transfer information. It also has only two gates, a reset gate, and an update gate. Besides, GRU unit controls the flow of information like the LSTM unit, but without having to use a memory unit. It just exposes the full hidden content without any control. Second, GRUs are relatively new, and their performance is on par with LSTMs, but computationally more efficient (they have a less complex structure). Finally, the third model uses two LSTM layers instead of one, and the last model has two dense layers in the output as classification heads. Table 1 and Table 2 show the results for different architectures on the validation and test sets respectively. Note that we ignore punctuation during the evaluation process.

Table 1: Accuracy of validation/test set for 50 epochs

	LSTM	**GRU**	**2x LSTM**	**2x Dense**
Validation set	82.92	82.66	81.00	79.73
Test set	83.74	83.23	81.71	80.57

Table 2: F1-score on validation/test set for 50 epochs

	LSTM	**GRU**	**2x LSTM**	**2x Dense**
Validation set	57.39	57.72	52.30	51.75
Test set	56.43	56.52	53.47	51.71

4 Discussion

The comparison of POS taggers is a delicate issue since many factors can affect the final accuracy figures: different sizes for training and test sets, different tag distributions, different tag sets, etc. In general, a major cause of POS-tagging errors is unknown words. Here, unknown words refer to those that have not appeared in the training data. It would often be difficult to recognize the POS label of an unknown word. A frequent source of unknown words in learner English is spelling errors. In our project, as can be seen in the previous sections, there is not much difference in the metrics over the whole dataset. However, by looking at the results related to every single class we can distinguish the ones that caused an increase in error and reductions in the quality of the metrics. For instance, the results of the model with GRU layer show that tags 'NNPS', 'WP$', 'PDT', and 'RB' which stand for Plural Nouns, Possessive Whpronouns, Predeterminers, and Adverbs respectively have the lowest f1-score. One possible reason can be due to the low number of support for these tags, for example, as can be seen in Fig. 1, 'WP$' has extremely low support in the train set. Also, the classification report shows that most of the 'NNPS' tags are classified as 'NNP' which may be due to the spelling or grammatical errors in the dataset. Another one may be related to OOV words. Since we used placeholders (vectors of zeros) for OOV words, the model may struggle for classifying them.

5 Conclusion

In this work, we have presented and evaluated a machine learning based algorithm for POS tagging. We experiment with different models on this task which have almost similar performances except for the model with multiple dense layers. We saw that GRU has slightly better performance than others and is faster.

References

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Lluis Marquez, Lluis Padro, and Horacio Rodriguez. 2000. A machine learning approach to pos tagging. *Machine Learning*, 39(1):59–91.