Università di Bologna - Master's degree in Artificial Intelligence

Assignment No. 1: recurrent neural models for sequence labeling

Marco Costante, Alex Costanzino, Alessandra Stramiglio, Xiaowei Wen

Abstract

Part-of-speech tagging (POS) is one of the most important classification task in natural language processing.

In this assignment we experimented a bidirectional model, along three variations, implemented with Tensorflow API, based on Keras backend, and tested the performances with different input data strategies and hyperparameters.

1 Data loading

We decided to feed the neural networks with sentences, instead of the entire documents, since the lengths of the documents have too variability with respect to the lengths of the sentences. In this way we have a better trade-off in the padding and truncation process.

We loaded the splits into separate dataframes, composed by three columns: `file_name`, `words`, `tags`.

2 Data inspection

For training, as maximum sequence length we considered *95th* percentile of the lengths of the sentences of the training set (66 tokens).

In our case we know that such length can be used also for testing, since the maximum length of the sentences of the test set is way 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.

3 Data processing

At first we created the vocabularies of words and tags for the training set, we extracted the out-of-vocabularies (OOV) words for it and then its embedding matrix. We assigned random vectors for the OOV words. We repeated the same process for the validation set, taking into account the fact that the OOV words already considered for the training set, are not OOV words for the validation set. The same was done for the test set.

Then, we concatenated all embedding matrixes into a single one, and merged the various vocabularies into a single one.

Finally, we put our dataframes into matrixes, padding and truncating the sentences, using the aforementioned vocabulary to encode the words into integers.

We also created a vector of weights for each POS tagging classes, to take into account the unbalance of the dataset, based on their inverse frequency. The weights are amplified with a constant factor to avoid too low loss function during training, since it would be more difficult to track.

4 Models and training

We created a baseline model, composed by three layers: a non-trainable embedding layer, that also takes into account the masking of the 0s used for padding, a bidirectional LSTM layer with a dense layer on top for classification. The dense layer has the same number of nodes of the considered tags plus one, since we reserved 0 for padding.

Then, we made some variations from the baseline model:

- Variation No. 1: replacement of LSTM layer with a GRU layer:
- Variation No. 2: addition of another LSTM layer.
- Variation No. 3: addition of another dense layer.

For embedding we decided to choose the maximum available dimension for GloVe (300) to have the best possible representation.

We performed a grid search between [16; 256], stepping with powers of two, to choose the number of hidden nodes of the LSTM layer: we obtained as best result 128 hidden nodes. We chose to dropout the 20% of nodes and the 10% of recurrent nodes of the LSTM layer.

We trained our networks with RMSprop optimizer, using a learning rate scheduler that decays exponentially the rate, which starts at 0.001. We also tried Adam optimizer, but adaptative methods seems to stall the learning.

As loss we used sparse categorical cross-entropy with softmax activation function for the last layer since we worked with integer labels. We also tried sparse categorical focal loss with class weights to enhance the dataset unbalance, but it turned out too difficult to optimize.

For the last variation we doubled the number of hidden nodes of the first dense layer.

To early stop the training we tracked the validation loss, with a patience of 5 epochs, since the validation accuracy also takes into account the classes that we would later discard for evaluation.

5 Results and error analysis

The best models are variation one and two.

Model	**Epochs**	**Validation accuracy**	**Test F1-score**
No. 1	27	0.8122	0.75
No. 2	20	0.8099	0.74

To calculate the F1-score we discarded the classes with null support in the test set. We could also consider their prediction 1 or 0, but we would overestimate or underestimate the score. The choice is application-dependent. This is also the reason why we obtained such low accuracy scores on the test set.

Moreover, since the distribution of the test set is different with respect to the training and validation sets, it is not representative, thus a more balanced dataset should be considered for evaluation of these models.

6 Future developments

As future developments we could use more deep models to enhance the representation capacity. Then, we could also explore other scheduling strategies for the learning rate and other optimizers as well. Also, a more representative test set could be used, to validate more accurately our models.