NLP: Movie Sentiment Classificator

Due on 1^{st} of April

Teacher: Leon Derczynski

Author: Evgeny Gryaznov

Assignment Description

In this assignment we were asked to do the following:

1) Write a movie review sentiment classifier using Keras LSTM implementation and IMDB data. This part requires us to understand Keras library's internals, in particular, the *layers* and how they can be combined with each other. Also we need to know how LSTM networks work.

2) Evaluate obtained model against the custom baseline. Here we just need to choose a threshold and show that the final accuracy of our LSTM on test data passes it.

3) Extend the model beyond the given specification and try to get a score over 92%. Here we decided to tackle the same problem but with a little twist: consider a new natural language.

Solution Examination

First Part: English Classifier

Jason Brownlee, in his online article [2] makes it crystal clear how to train an LSTM neural network in Python to solve a movie review classification problem. Thus, all we need to do is to follow his conscience lead and make sense of it all on the way. The IMDB dataset is already there, just waiting to be used. But it would be a pity to just copy/paste his work and made it look like ours, so we wrote a jupyter notebook which explains Brownlee's code and shows that we have a good understanding of what's going on there, also it's intended as a continuation of this report. The actual implementation of our LSTM is contained in the python script *"english-classifier.py"*, but you can also run the model inside the notebook.

For the evaluation of the first part, we've chosen the following baseline: 80% of the final accuracy of the test data. Once executed, our LSTM should display at least 86%

To learn more about our English classifier, please look into the notebook.

Second part: Russian Classifier

For an extension, we decided to do the same thing, but with the Russian, because it's interesting to work with our native language in this course. New natural language means that we would need our own dataset. After excessive googling and a lot of reading, we came to the conclusion that there is nothing like the IMDB dataset for Russian, so we decided to find a web-site like *"rottentomatoes.com"* such that:

1) Contains a lot of passable-quality movie reviews together with their ratings.

2) Its HTML structure is relatively easy to parse.

Luckily, one day we stumbled upon the site [1] which turns out to be a gold mine for us! We wrote a small python module, *"ru_otzyv"*, that crawls through and parses all the reviews that were submitted. Secondly, we did the same thing with this data that the authors of IMDB dataset did:

1) Generated a vocabulary of all tokens that appeared in the set.

2) Sorted them by inverted frequency.

3) Replaced each token in each review by its position in top frequency rank.

4) Polarized the ratings: if a rating is ≤ 4, it's bad. Ratings with 5 were dropped because they are too ambiguous. Consequently, a rating is considered to be good if it's ≥ 6. The maximum is 10 stars.

5) In the process, we discovered that ratings were distributed unevenly.

6) Split the data into a train (70%) and a test (30%) part.

Now we got three files, each of them stores its part of the dataset:

- *"cooked-dataset.json"* contains the final, ready-to-use version of the data.
- *"ru-reviews.json"* contains the actual text of the parsed reviews.
- *"ru-vocab.json"* is a vocabulary of all tokens that were extracted from the reviews.

The dataset manipulation is done via the *"ru_otzyv"* module, which has all the functions you will need. For example, now there might be new reviews available at [1], so we can update our data by issuing:

```
import ru_otzyv as ru
ru.update(ru.load('ru-reviews.json'))
```

There are plenty of other examples in the jupyter notebook which we created specifically for explaining what we did in the second part.

The actual LSTM implementation is also quite different from the original in [2]. We introduced a *dropout* technique to avoid overfitting and increase learning speed. We were able to get the 94% accuracy on the training data, 84% on the validation data, and 86% on the test data. Although last two are certainly not above desired 92 percent, but since it's custom data we don't know it it's possible to achieve 95% on everything without overfitting or cheating.

Lastly, we need to mention two things. Firstly, all the fancy plots and figures are in the notebook and can be generated, so you can ensure the validity of them. Lastly, we included a `predict()` method into our module and a couple of high-quality reviews so you can easily pay around with final model, have fun!

References

[1] *Russian Movie Reviews* http://kino.otzyv.ru

[2] Jason Brownlee, *Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras*, http://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/ 2016.