Sentiment analysis using Yelp Reviews

Mårten Nilsson, Axel Demborg, Margaux Ledieu

January 9, 2018

1 Introduction

The aim of this project was to perform sentiment analysis on reviews from the Yelp Reviews dataset by predicting the amount of stars awarded from the text of the review. The end goal of this project was to implement live sentiment and quality analysis of written text for helpful feedback to reviewers.

2 Data

The dataset used for this project is the *Yelp Reviews* [1] dataset. The dataset contains 4.7 million reviews of business from all over the globe. Each review consists of the written text, the amount of stars awarded as a integer, one to five, and the number of votes the review got from other users for being *useful*, *cool* or *funny*. There is also some meta data like the date the review was submitted and unique ID:s for the user, business and review itself.

The review data set was randomly split up into a training set (80%), a validation set (10%) and a test set (10%).

3 Implementation details

The project was implemented in *Python* using the *PyTorch* deep learning library [2]. The code is available here.

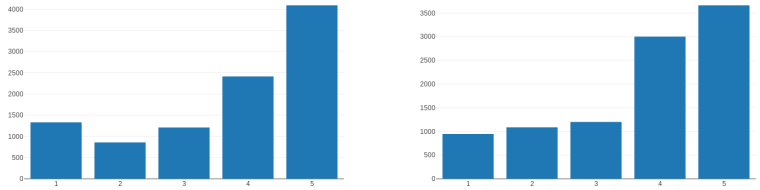
4 Models

For prediction a few different models were tested. All models were based on recurrent neural networks and was designed to handle arbitrarily sized sequential data. The specific models evaluated in this project are:

- **Baseline RNN** This model was built to be the simplest possible as a test of our training framework. It is a single layer Elman RNN [3] with 100 hidden nodes and a ReLU output layer.
- **LSTM** This model consisted of a single LSTM [4] cell and a ReLU output layer.
- **GRU** This model is the same as the LSTM model but with an GRU [5] cell instead of the LSTM cell.
- **CNN-LSTM** This model keeps the same LSTM layer, and adds a convolutional layer before.

The models were developed in two different types. The first type assumed the input data in the form of preprocessed vectors, the second type assumed the sequences to be sequences of indexes and was responsible for learning it's own representation of the entities of the sequence. The models themselves are agnostic towards character-based and word-based data. This allows the same code to be used in a variety of settings when experimenting with sequential predictions.

After performing some small preliminary experiments it was decided that the experiments in this project would focus on word-based models due to computational efficiency. To maximize the performance of the models pre-trained word vectors instead of letting the models learn their own representation of the words.



(a) Real (b) Learned

Figure 1: Star distribution for the real data and the one learned from the LSTM model. The distribution was estimated during training using an exponential window of star counts (y-axis).

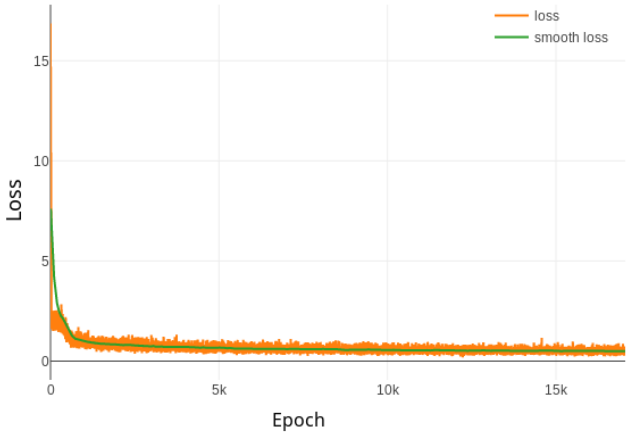


Figure 2: Loss during one epoch of training for the LSTM model

5 Performance and Scalability

The base models required around ten hours of training per epoch on the training set. Using an GTX660 Ti GPU this time was reduced by a factor 10, making training of the models feasible.

The data sets were stored in json format. Instead of preprocessing the data sets, the data processing was performed on the fly during training on a separate thread pool. This was performed through the built in *DataLoader* class in *PyTorch*. As long as the forward and backward pass are slower than the data processing this does not slow down the training of the models.

The data was read randomly using an indexed memory map to rapidly access the individual reviews in $\mathcal{O}(1)$ time even though the data is stored in a single json file. This method scales to huge files as long as the line indexes of the files fit in the memory of the computer reducing the need of extra data processing.

6 Results

Each of the presented models were trained for a single epoch. Thereafter the trained models were evaluated by recording the mean square error (*MSE*) of the predictions of the models on the test data set. All models were trained with and Adam optimizer [6], with a batch size of 100 and a hidden state of size 128 except for the CNN-LSTM which instead used a deeper architecture and had a hidden size of 64.

Results from these experiments are presented as the *MSE* on the test data for the different models after training in Table 1. The loss plot from one epoch of training for the LSTM model can be seen in Figure 2 and a comparison between the learned and the real star distribution can be seen n Figure 1.

Table 1: Mean squared errors on test set for the different trained models.

Model	MSE on test set
RNN	1.929
LSTM	0.418
GRU	0.415
CNN-LSTM	0.451

7 Discussion

From the results we can see that both the GRU model and the LSTM model outperform the Elman RNN in our experiments. This is reasonable since both the LSTM and GRU are more sophisticated models capable of better capturing long-term dependencies. What is interesting is that the GRU displays slightly better results. The difference is hardly significant but since the LSTM model is more complex than the GRU model one would expect it to outperform the GRU in this setting.

It is interesting to note that the real star distribution is somewhat convex and biased towards high ratings. The learned distribution on the other hand is concave which is closer to the properties of a bell curve. A possible explanation for this is that convex shapes are more complex to model and require either larger hidden sizes or deeper structures than what can be captured by the current models.

Another possible cause of the different predicted distribution is that the data set does not exclusively contain English reviews, though the word vectors used are English. In the case of an unrecognized word the data processor will convert it to the zero vector. An foreign review might contain a large number of unrecognized words which then presents the model with very little information. Since the model is deterministic it is not unreasonable to guess that these reviews are mapped to a single mean score around 4, therefore skewing the distribution.

References

[1] Yelp dataset. https://www.yelp.com/dataset. Accessed: 2017-12-12.

[2] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017.

[3] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.