Deep Learning – Final Project (Parameters)

Max Velich (s2762927)
Eline Meijer (s2478889)
Menno Liefstingh(s2735059)
Niek Naber (s2515970)

May 8th 2020

Software and Hardware Both models were built in Python 3.7.4 (Van Rossum & Drake, 2009) using Tensorflow 2.1.0 (Abadi et al., 2015) and Keras 2.3.1 (Chollet et al., 2015). The models were trained on Peregrine, the high performance computing cluster of the University of Groningen. We've used a grid of V100D-32Q nodes to run our models on Peregrine. The V100D-32Q nodes consist each of 80 multiprocessors with 60 cores and runs with a clock-rate of 1.38 ghz. The Tensorflow/2.1.0-fosscuda-2019b-Python-3.7.4 version was used on the peregrine cluster, which took care of the parallel execution of our code. The LSTM model took roughly 100h to train, while the recurrent neural network took about 50h to train, both for the versions we describe in the report.
For the sake of readability and clarity we converted the Python files to Python notebooks. Some comments are added to help guide anyone who needs to read them through the code.
We used snippets and understanding from the following website:

- Tutorial on Text Generation with LSTM: `https://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/`
- Tutorial on Generator: `https://medium.com/analytics-vidhya/train-keras-model-with-large-dataset-batch-training-6b3099fdf366`
- Fixing a bug in a Keras Tokenizer: `https://github.com/keras-team/keras/issues/8092`

The code from the first source was used as a guideline. We used a different data set as in this code. Because of that, the code needed to be adjusted. The preprocessing was completely different, and the code needed to contain a generator like the one from the second source. The generator needed to be adjusted a lot to fit our needs, only the general concept was taken from the source. We also played around with the number of epochs that we needed to use. Additionally, we added an extra generation method that was based on sampling from a probability distribution.
Next to this, we also implemented this pipeline with a recurrent neural network so that we could compare it to a simpler method. To do this a lot of dimensions and data structures needed to be adjusted as compared to the LSTM code file. For the RNN we also implemented the shrunken vocabulary size. The actual Keras Tokenizer should be able to do that for us, but that code apparently contains a bug. The third source told us how to fix this bug.

Data set We used a data set provided by Amazon (*Amazon Customer Reviews Dataset*, 2015). It consists a set of customers' product reviews for a specific category. For training we used the subset of reviews of digital products. For testing we used the subset of reviews of video games.

As we developed our models, an obstacle was the time consuming training of our models. The focus of our adjustments to the models were therefore on the downsizing and accelerate the training time, without getting the performance of the models deteriorated.

Individual contributions We split the work into the following categories and tried to uniformly work together as much as possible.

- Data Preparation: Niek & Max
- RNN: Niek & Max
- LSTM: Niek, Menno & Eline
- Experiments: Niek, Menno & Eline
- Report – Introduction: Max & Eline
- Report – Methods: Niek, Eline & Max
- Report – Results: Niek & Menno
- Report – Discussion: Max, Niek & Menno

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Amazon Customer Reviews Dataset. (2015). `https://s3.amazonaws.com/amazon-reviews-pds`. (Accessed: 2020-05-04)

Chollet, F., et al. (2015). *Keras.* `https://keras.io`.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual.* Scotts Valley, CA: CreateSpace.