Deep Learning – Final Project

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

May 8, 2020

Abstract

In this paper, two text generation models are compared that are trained on Amazon's customer reviews for digital products. The first model is a traditional recurrent neural network (RNN), while the second model uses a modern Long-Short Term Memory (LSTM) system that includes an embedding layer. Text sequences are generated by feeding the start of new video game reviews to the models. Results are evaluated by a subjective qualitative analysis. Despite the fact that some parts of sentences were often repeated in text generation, the LSTM model outperforms the RNN. The syntactic structure of the text generated by the LSTM model is more consistent than the text generated by the RNN. Although the necessary parameter tuning to restrict training time made direct comparison rather difficult and might have resulted in sub-optimal results. We demonstrate promising results especially for the use of LSTM models in text generation tasks.

Introduction

In this paper we aim to compare an old method of computational language simulation with a more modern model. We are feeding text into a recurrent neural network and generate a prediction based on that. We use the same input with a Long-Short Term Memory model with an Embedding layer to compare the results to. We are using the open data set of customer reviews provided by Amazon (*Amazon Customer Reviews Dataset*, 2015). The results are evaluated in the light of human comprehension and the emergence of grammar.

Background

First, we want to give a brief outline of papers that used different versions of neural networks with language inputs in the past. We will show how Elman (1990) used simple recurrent neural networks to show the non-deterministic behaviour that is necessary to produce language. This then got picked up by Mirman, Graf Estes, and Magnuson (2010) who showed that a recurrent neural network shows robustness against ungrammatical inputs given a certain window of grammatical inputs prior. Lastly, we show that LSTMs overcome the vanishing or exploding error problem of recurrent neural networks, and therefore provide a more modern solution.

Finding structure in time Computational systems generally used to be of deterministic nature – a given unique input produces a desired unique output. Language is nondeterministic. That means that a given input can result in many different outputs. However, there are more or less strict grammar rules in language, which can be learned by humans fairly quickly. Teaching a neural network this nondeterministic behaviour was successfully tackled by Elman (1990) who used a simple recurrent neural network: a normal neural network with a hidden context layer that acts as a feedback loop to the hidden units. This is meant to simulate the "recall" of the previous step, enabling a temporal aspect of neural networks. This was done since it represents the nature of language more accurately – humans perceived words one after the other, and not a full sentence in one go. The goal of Elman (1990) was to let the network develop internal representations about the order of the inputs, thereby *finding structure*

in time. The network showed many signs of success, including learning generalizations like parts of speech and semantic classes. In fact, according to (Elman, 1990), the network was able to find hidden unit representations that group words into nouns and verbs. Albeit not without criticism, this paper has been shown to find generalizations of a nondeterministic temporal input such as language.

Grammar over frequency The study by Mirman et al. (2010) builds on the previous paper's findings, specifically on the property that simple recurrent neural networks can become sensitive to boundaries between words which are inferred from transitional probabilities of the input. They showed that recurrent neural networks elicit statistical feature of human word learning, frequency of syllable parts dominate in the beginning but is replaced by a transitional probability as time moves on. That means that while the network is learning just the most common syllables in the beginning, the underlying rules of language, i.e. grammar, are slowly encoded in the hidden neurons. They conclude that their performed simulations reveal recurrent neural networks as a possible candidate to help understand the link between human word learning and statistical learning processes. Given both the older account of Elman (1990) and this new account of Mirman et al. (2010), we take a recurrent neural network as our baseline for a language learning comparison in this report.

Long-Short Term Memory A more modern alternative to recurrent neural networks is provided by Hochreiter and Schmidhuber (1997). They explain the problem of rapidly increasing or quickly vanishing error signals in recurrent neural networks. Depending on the magnitude of the weights in the network their errors exponentially evolve with time. This can be useful as it enables a short-term memory behaviour of the networks as was explained in the previous papers. The solution provided by Hochreiter and Schmidhuber (1997) is the long-short term memory model (LSTM). It is designed to maintain the short-term memory behaviour while overcoming the exploding-in-size errors over many steps during the training phase. This works by enforcing constant error signals. This has been shown to provide generally better results for word labelling tasks (Yao et al., 2014) and language modelling (Sundermeyer, Schlüter, & Ney, 2012) over conventional recurrent neural networks.

Before using the LSTM, we will implement an embedding layer that represents the words in the form of distributed vectors (Bengio, Ducharme, Vincent, & Jauvin, 2003). This results in a vocabulary matrix that contains the mappings from the words to real-valued numbers. This is necessary as words are not numbers (and networks only understand numbers), but also to reduce the number of the input neurons for the LSTM. This stands in contrast to the one-hot encodings that both simulations of Elman (1990) and Mirman et al. (2010) used. We do not include embedding layers in the recurrent neural network account as embedding layers are fairly modern, and we want to compare the older approaches with newer ones.

Research Question

Given that previous models have been shown to generate emerging grammar with a degree of success, we ask the question if a modern LSTM model (with an embedding layer) can still reproduce such results. Specifically, does a LSTM model produce better results than a simple recurrent neural network? In this report we aim to predict text based on two architectures and then comparing those predictions.

Given the more efficient nature of the LSTM we assume that they can deal better with big data sets (i.e. learning over time), while the older models will show better non-deterministic behaviour of language generation (i.e. inferring grammar). Therefore, we believe that the LSTM will be a better solution for large amounts of text, but that the RNNs will reproduce the grammar better.

The details of the training process are outline in the next section. After training we will use the same test inputs for both models, and compare the predictions to evaluate the models. The evaluation process is also outlined in the following sections.

Methods

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. To limit the size of this data set, only the *digital software* reviews are used for training the network. For the generation of new reviews

we will later use the *video game* reviews as a test set. For our purposes we used the field *review_body* which contained the actual reviews that customers left for a specific product.

Preprocessing To preprocess the data, we first removed all punctuation and converted the text into lower-case letters. As we are primarily interested in an emergent learning of grammar, punctuation as well as the casing of the letters is irrelevant. We then prepared the data set in the form of one long consecutive string of words. Then we used a tokenizer as implemented in Keras (Chollet et al., 2015) to map each word in the input data to a single integer. All words that are mapped to an integer in this process belong to what we call the vocabulary. The vocabulary size is therefore the number of different words that we encountered during this process. To understand the reasoning behind the remaining preprocessing and training steps, we first have to describe the structures of the models. In the next paragraphs, we will first describe the model structures, after which we will continue describing the remaining preprocessing and training steps.

Recurrent Neural Network The recurrent neural network has the vocabulary size as its input and output size, because this network uses one-hot-encodings of words. The model consists of a hidden simpleRNN layer with hyperbolic tangent activation functions of size 100, and a time distributed Dense layer with a softmax activation function. The number of trainable weights is mainly dependent on the vocabulary size. Because a high number of trainable weights means a slow training process, we needed to reduce this number. According to Zipf's law (Zipf, 1999, 1949), the frequency with which a word occurs in a language, is inversely proportional to the rank of that word in its frequency table. Fagan and Gençay (2010) mention an example of this: only 135 words are needed to already make up 50 percent of all of the text in the Brown Corpus of American English text. Taking this into consideration, reducing our vocabulary by 80 percent is a reasonable choice. We reduced the vocabulary size from the original 49253 words that are encountered in the digital software dataset, to only the 10000 most occuring words in the digital software dataset. We calculated from the data set that the top 10000 words make up 98.8 percent of the text. The input and output layer now consist of only 10000 nodes instead of the original vocabulary size.

After the previously described preprocessing steps, we have one long consecutive list of integers, where each integers number represents a unique word. Starting at the first word, the words are all iteratively fed to the network as one-hot-encodings of size 10001, with the next word in the consecutive list of integers as its successor. If a word is not in the vocabulary of the network, it will be mapped to the one-hot-encoding of integer 10001. During one epoch of training, all words have been presented once. The network will be trained for 50 epochs in total.

Long-Short Term Memory The LSTM has a different input size from the RNN, because the first layer of the model is an embedding layer. The embedding layer takes care of mapping integers to word-vector representations. Therefore, no one-hot-encodings, but sequences of integers are needed as input. The input consists of 19 integer numbers. The model first consists of the embedding layer of 100 units, then two LSTM layers of 100 units. After that, the model has a Dense layer of 100 units, with ReLU as its activation function. A dropout rate of 10 percent is applied to the output of the Dense layer. This is in turn fed to the last Dense layer. This layer has a softmax activation, and the total vocabulary size as its size. Note that here we used the whole vocabulary because this model can handle a bigger vocabulary size in the same training time.

This model uses the same list of integers to train on. As many different sentences are made with 20 consecutive numbers in this list as possible. During one epoch of training, the network encounters every sentence once. The first 19 integers are the integer list. The last integer is mapped to a one-hot-encoding of that integer and is used as the successor. The network will be trained for 200 epochs in total. The difference in number of epochs compared to the RNN can be explained because the RNN needs to learn less due to its smaller vocabulary size.

Evaluation Evaluating text generated by automated means is inherently difficult due to its nondeterministic nature. There are evaluation metrics like the BLEU score (Papineni, Roukos, Ward, & Zhu, 2002) that are meant for machine translation, but the performance of text generation is hard to measure. We can, however, make a subjective qualitative analysis of text generated by our models. We are mainly

interested in seeing if the more modern LSTM can outperform a classic recurrent neural network. This is something that can be assessed qualitatively with our understanding of language.

Results

To generate new reviews with the trained models, we need to supply the models with input strings. The specific input format differs per model. The LSTM expects a sentence of 19 word-to-integer mappings, while the RNN expects one one-hot-encoded word at a time. To have similar inputs in the test set as in the training set, we will use the video game review dataset of Amazon to supply as an input to our models (*Amazon Customer Reviews Dataset*, 2015). The first 19 words of these reviews are fed to the models, to generate the next 25 generated words of a possible review. Elman (1990) writes, we should remember that a prediction task is nondeterministic and successors cannot be predicted with absolute certainty. We therefore do not aim at generating the exact same reviews as the reviews in the new test set. But, as Elman (1990) also says, word order is not random in a generation task. For any given sequence of words there are a limited number of possible successors. We will therefore mainly look at the structure of the generated part of the review, instead of the exact words used.

The RNN is fed one one-hot-encoded word per time step. So when the next 25 words of a review are generated, the first 19 are fed to the model first. Every word in this input sequence is first mapped to the right one-hot-encoding, that matches with the one-hot-encodings during the training process. It should be noted that since we needed to restrict the vocabulary size, not all words in the test set are in the vocabulary of the model. Just as during the training phase, words that are outside of the vocabulary are just skipped.

The (at most) 19 one-hot-encodings of the input sequence are sequentially fed to the recurrent neural network. Only in the last iteration of feeding these input vectors to the network, the output becomes interesting. The output of the 19th input vector, is a probability vector of size 10000. To convert this to a word we use two different techniques. We take the argmax of this distribution, as well as a random sample over this distribution. The value that is gotten is converted to a word and its on-hot-encoding vector is again fed to the network to predict the next word. This process goes on until 25 new words are predicted.

The LSTM is fed a list of 19 integers, that correspond with the words mapped to the integers as determined in the preprocessing / training phase. Then, the LSTM outputs a probability distribution of size equal to the vocabulary size. Again the conversion to a word is done via the two previously described techniques: argmax and probability distribution sampling. The obtained word is mapped to its corresponding integer and added to the 19 integer list. The first integer of this list is dropped. The new list is used as input for the LSTM again. This process is repeated 25 times to generate 25 new words that continue the 19-word input sentence.

We will now provide some sample texts generated by the models and qualitatively analyze how well they resemble text written by a human. The first input for the models is:

```
Simcity cannot work on dell laptops with switchable graphics using AMD GPU (HD7730
    with Intel HD 4000). Such configuration
```

The output of the LSTM and RNN for the argmax and probability distribution sampling respectively is:

```
RNN (Argmax): ...  such configuration av's long teenage shields wife's pops rollback
    outsourced shift several pluses pirated ripping na overseas 82 dilemma butt
    applies displayed extensions

RNN (PDS): ... such configuration subjected attention enhancing cc whether thai aged
    pcs precaution riddle definitive protector mcafee gadget section combination
    suffer designate motion renewal tutor barcode massachusetts senior pal shove

STM (Argmax): ... such configuration and the new fonts were not as a uniform and they
     were not a embeds that i could see are the same to be transcribed

LSTM (PDS): ... such configuration is much more advanced among large limitations
    sometimes it takes most advanced processes it simply looks but apparently it's
    doing a way to control your
```

The LSTM, however, does output some interesting sentence structures. Although the sentence is gibberish, we see grammatically correct structures of up to 5-6 words in length emerging from the output.

This is especially true for the Argmax conversion, where we see structures like *'the new fonts were not as uniform'*, a phrase that contains correct usage of verbs, nouns and an article in a structure spanning 7 words. Another example is given by the input:

```
Having grown up with the franchise, SimCity has a special place in my heart. I spent countless hours with
```

This resulted in the following outputs:

```
RNN (Argmax): ... countless hours with appreciated intruders reported child's obscure catching piano certificates virtualization blurry endured ver connectify phoenix headed flagship gtx terrible serious assigning ruler

RNN (PDS): ...countless hours with animation lease summaries spyware footage neglected weakness centered darn mp3s covering event november connected dining scripting 760 advertised booster nearly 49 revamped payees recreate recreate warranty

LSTM (Argmax):} ... countless hours with the software and i have been using it for years and i have been using it for years and i have never had a problem

LSTM (PDS): ...countless hours with the new program the mac version that i have originally to use a more terrible program i had used my password theft on line this
```

While the performance of the LSTM in combination with the argmax conversion to words seems relatively good, there is the problem of some phrases and nouns that reappear in more than half of the outputs of the model. Some examples of these:

```
LSTM (Argmax): ... playing it's a great product i have been using quicken for years and have never had a problem with it i have been using turbo tax for

LSTM (Argmax): ... we have been using it for years and i have never had a problem since i purchased it and i was able to download my old version

LSTM (Argmax): ... so why would i buy this product i have been using turbo tax for years and have been very satisfied with the product i have been using turbo tax for years
```

Discussion

For the sake of brevity we are not able to include all our outputs in this report. However, we will try to list the main takeaways from going through all the output from the models. First of all, the recurrent neural network is unable to come up with any output other than a seemingly random string of words. Perhaps a more sophisticated approach or more training would allow the RNN to produce better results, but this seems unlikely as it does not appear to have learned anything worthwhile in the first 65 epochs.

The LSTM, especially paired with the Argmax conversion to words, is able to produce some promising grammatical structures spanning sequences of up to 10 words long. The only caveat to this is that it also often produces repetitions of the same few strings, or slight variations on those strings. The most notable example is "I have been using it for years" and "easy to use". Whether the frequency of these strings stems from the strings being grossly overrepresented in the training data is not plausible, but it is more likely that a slight overrepresentation of these phrases is magnified greatly by the model. Another noticeable result is that some nouns are very often present in the generated text, most notably "quicken" and "avast" which are both types of software. The occurrence of these nouns often goes hand in hand with the repeated sentence structures that we detailed above.

Conclusion Given our evaluation criteria outline in the Methods section, and our results from both models, we conclude that the long-short term memory model (with embedding layer) indeed outperforms the recurrent neural network. We base this mostly on subjective matters as predicting text accurately is fundamentally impossible given its nondeterministic nature.

The objective of our research was to find out whether the LSTM performs better in text generation than the classic recurrent neural network. Based on the output from the different models, we can be sure to conclude that. The LSTM paired with the Argmax conversion was able to produce some grammatical

structures vaguely reminiscent of human-written text, although we saw definite flaws in the frequent repetition of some sentence structures and the over-representation of some nouns.

Critique We used different input transforms (one-hot encoding and embedding layer) for both models which might results in a disadvantage of the recurrent neural network. We chose to do this given a historical context, and our aim to compare the old methods to the new. However, a third model that uses recurrent neural networks in combination with a embedding model seems appropriate now.

Due to time issues we were not able to train the RNN model for more than 65 epochs. The LSTM ran for more than thrice as many epochs as the RNN. Despite the fact that the RNN did have a smaller vocabulary and therefore needed to learn less, it could still be that the difference in epochs or the difference in vocabulary size gives an unfair advantage to either one of the models. We would recommend repeating the experiments with equal vocabulary sizes, and an equal number of training epochs.

The data set we used is sub-par in terms of grammatical accuracy and English language use overall. Reviews that are left on customer portals like on Amazon's website are riddled with mistakes, typos and rushed comments. As a consequence we had to discard many words that were misspelled which muddied the statistical features of the data set. For instance, if the word 'word' was misspelled as 'wrod', the frequency of the word 'word' is now inaccurate since we do not correct for misspellings.

Given the size of our network it made direct inspection of the hidden layers nearly impossible. That meant that we could not directly observe the trained weights of the individual neurons and how they might relate to a generalization of grammar, such as was done by Mirman et al. (2010). This means that we cannot say with certainty that the networks correctly generalized grammar rules from the input. Ultimately, the given input might have been too complex for the recurrent neural network.

Future Research For future studies we suggest first using a clean corpus of text. This has the advantage that one can feed the neural networks with actual high-quality inputs. This stands in contrast to the reviews provided by Amazon customers. Although, remarkably, the networks show a certain degree of tolerance towards ungrammatical input. This is inline with the studies of Misyak, Christiansen, and Bruce Tomblin (2010). They introduced an entirely ungrammatical block of inputs after a training phase with a grammatical block that was six times longer than the ungrammatical one. The results show a fairly quick recovery when again exposed with a grammatical block. We observe similar features, and recommend not to shy away from flawed input when dealing with language prediction.

On the basis of Zipf's law we excluded about 80% of unique words from our dictionary. This was done to increase performance of the training. Although, it sounds drastic to exclude that many unique words, it only accounts for roughly 1.2% of the total words that were used for training. Languages feature redundancy of words to overcome perception and comprehension errors. While humans benefit from that, the clear cut mapping to real-valued numbers is not prone to misunderstandings. Ergo, to infer grammar and underlying statistical distribution of the input, we advocate to experiment with more drastic reduction of the vocabulary, to find an optimal ratio of performance to accuracy. Other techniques that might be helpful here are the replacing of misspelled words with their correct forms as part of preprocessing, or replacing uncommon words with their synonyms.

References

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

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, *3*(Feb), 1137–1155.

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

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science TA - TT -*, *14*(2), 179–211. doi: 10.1207/s15516709cog1402_1 LK - https://rug.on.worldcat.org/oclc/5155702770

Fagan, S., & Gençay, R. (2010, 12). An introduction to textual econometrics. *Handbook of empirical economics and finance*, 133-154. doi: 10.1201/b10440-6

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. LK - https://rug.on.worldcat.org/oclc/120718902. *Neural computation TA - TT -*, *9*(8), 1735–1780.

Mirman, D., Graf Estes, K., & Magnuson, J. S. (2010). Computational Modeling of Statistical Learning: Effects of Transitional Probability Versus Frequency and Links to Word Learning COMPUTATIONAL MODELING OF STATISTICAL LEARNING. *Infancy TA - TT -*, *15*(5), 471–486. doi: 10.1111/j.1532-7078.2009.00023.x LK - https://rug.on.worldcat.org/oclc/4650254813

Misyak, J. B., Christiansen, M. H., & Bruce Tomblin, J. (2010). Sequential expectations: The role of prediction-based learning in language. *Topics in Cognitive Science*, *2*(1), 138–153.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics* (p. 311–318). USA: Association for Computational Linguistics. Retrieved from https://doi.org/10.3115/1073083.1073135 doi: 10.3115/1073083.1073135

Sundermeyer, M., Schlüter, R., & Ney, H. (2012). Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association.*

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., & Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *2014 ieee spoken language technology workshop (slt)* (pp. 189–194).

Zipf, G. (1949). *Human behavior and the principle of least effort: An introduction to human ecology*. Addison-Wesley Press. Retrieved from https://books.google.nl/books?id=1tx9AAAAIAAJ

Zipf, G. (1999). *The psycho-biology of language: An introduction to dynamic philology*. Routledge. Retrieved from https://books.google.nl/books?id=w1Z4Aq-5sWMC