NATURAL LANGUAGE PROCESSING - THE ONE WITH FRIENDS

Daniel Juhász Vigild (s161749), Jonas Søbro Christophersen (s153232) & Lau Johansson (s164512)

Abstract

In this paper we construct a language model for text classification and generation of dialogues from the TV series *Friends*. Utilizing state-of-the-art methods to improve performance, we build a Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) architecture. Through tuning of hyperparameters and regularization using variational sequence length, DropConnect and Dropout we achieve a perplexity of 92.3 on the Penn Treebank data set. We then utilize the found model to generate dialogue for the TV-series *Friends*. Prior to this, we pretrain the model on Wikipedia and *Seinfeld* text, where it is shown that pretraining on *Seinfeld* improves performance on *Friends* text classification. Finally, we visualize the found semantics of *Friends* through t-SNE on the embedding weights and show a subset of the predicted dialogue. Data sets and code is publicly available at [5].

1. INTRODUCTION

The purpose of this paper is to build a neural network that can generate new dialogue for the TV-series *Friends*. The main reason behind this is to explore the boundaries of what neural networks can do. In other domains, for example hand-written digit recognition, neural networks have achieved near-human performance, but even when assessing state-of-the-art results within natural language processing, the generated text still have "traces" of a robots touch, and do not feel very natural or human-like. We find this to be a motivating challenge.

Generating new unique text that fits within a delimited semantic space - like the language used in *Friends* - is, apart from amusing for the content hungry fans of the show, also a commercially viable product. The technology can be utilized in next-word generation within message user interfaces (text-message, email, word processing programs) or entire sentences and conversations for chatbots in customer services.

This paper will start by building a model on the Penn Treebank data set (PTB) and thereby creating a proof-of-concept model. After this we will shift to the *Friends* data set and further tune the model. Here we will use pretraining on Wikipedia text (Wikitext-2 [3]) and dialogue from *Seinfeld*, to further enhance the general semantic understanding of the model. Finally, we will visualize learned semantic structures through t-SNE on the embedding weights and generate new dialogue for *Friends*.

2. MODEL ARCHITECTURE

In this section we briefly account for the use of the chosen architecture, a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM).

2.1 RNN

We build a recurrent neural network since it works well with sequences of data. Problems well suited for these types of networks would have data where a given observation is influenced by previous observations, for example time series data, or as in our case: the natural language from *Friends* dialogue.

2.2 LSTM

We use the Long Short-Term Memory because of its ability to retain information throughout the sequence, that would otherwise be lost with alternative methods. This is important, since dialogue might be influenced by information from the very beginning of a given sequence - say if something was briefly mentioned in the beginning of a season and then referenced to later in a different episode.

Furthermore, dialogue might have features that are constant over time (say if different characters have distinct ways of speaking) which are meaningful to retain, while some local trends might be important to address as well (say if a character is particularly agitated during an episode).

The LSTM enables both types of modelling by processing information through input, output and forget gates, and thereby output new cell states and hidden states, which are less prone to forgetting long term dependencies opposed to

regular RNN networks. The hidden state are used for the next word prediction and the long term memory is held in the cell state.

2.3 Performance measure: Perplexity

To evaluate model performance, we will use perplexity as a measure. This is aligned with the state-of-the-art methodology of the field.

2.4 Data

We begin modelling on the PTB data set. This is typically done for proof-of-concept of natural language processing models, and is also used for benchmarking. Our goal will be to achieve a perplexity < 100 for a model on the PTB data set, and then shift to the *Friends* data set for further modelling with the goal of generating coherent new dialogue.

3 . BUILDING THE MODEL ON PENN TREE BANK

In this section we explain the modelling strategy, and how we through adding complexity and regularization attempt to improve the generalization of the model. For each model, the specific values for all hyperparameters can be found in the results section, while the choices made and directions in which we changed important hyperparameters are described below.

3.1 Vanilla Model

We begin our modelling with a simple starting point with minimal complexity, no regularization and little thought into the specific values. This serves primarily as a reference point and a way to illustrate the effect of added complexity and regularization.

3.2 Tuned hyperparameters

Next we tune our hyperparameters, to achieve the best possible performance. We perform grid search by in concert trying different combinations of hyperparameters and attempt to find the combination that lowers perplexity as much as possible.

While optimizing hyperparameters we also attempted to change the optimizer, in our case from AdamW to (A)SGD. To this end we use a learning rate scheduler - since the optimization methods require different levels of learning rate. For this purpose, a Slanted Triangular, and a customized scheduler was trialed. Despite achieving a model that converged as fast as when using AdamW, we did not find any model where (A)SGD outperformed AdamW. The schedulers did at no point improve performance of AdamW, and was thus not used in any of the resulting models.

3.3 Regularization

To improve the performance of the model we introduced several regularizations techniques as proposed in [4].

3.3.1 Dropout
We used dropout on the embedding layer. This regularizes by randomly removing some of the nodes in the network so it does not become overdependent on a particular set of node connections - and thereby increases generalizability.

3.3.2 DropConnect
We used DropConnect in the hidden-to-hidden state weights, and tried different values before settling on the one which improved performance the most. DropConnect works similarly as normal dropout, but only removes weights rather than entire nodes. This is useful for time-dependent observations to ensure that impactful past information is not suddenly completely forgotten.

3.3.3 Variable sequence length
Finally, we used variable sequence length, randomly drawing each sequence from a normal distribution with a standard deviation 16 and mean 32. This ensures that the network does not only learn on a specific sequence length, but rather on both longer and shorter sequences - which enables different types of learning, that are to be used for different types of sentence predictions.

4. FITTING THE MODEL TO FRIENDS

Now we shift data set and begin modelling on *Friends* data. We describe how we used pretraining on Wikipedia and *Seinfeld* to improve model performance.

4.1 Pretraining

Transfer learning has shown big impact on performance in Computer Vision algorithms, and has recently been incorporated in NLP problems with succes. [1]

We attempt to improve our model performance by pretraining the model on a third data source before applying the model to *Friends*.
We have an assumption that more data is better and that pretraining the model on data from a different domain might improve the generalization of the model. In technical terms, we run the model on some text corpus, and save the network states. These states give us a “warm start” when we shift to *Friends* data, since the parameters of the model are not

initialized randomly but inherited from the pretrained model.

We will perform pretraining on Wikipedia text and on dialogue from the TV-series *Seinfeld*.

The Wikipedia text is contained as the Wikitext-2 data set, and the *Seinfeld* data contains dialogue from all episodes. All data sets (including *Friends*) are partitioned into 80% training, 10% validation and 10% testing splits. With 10 *Friends* seasons, and 9 *Seinfeld* seasons, the splits allot approximately one season for validation, and one for testing, with the remaining for training. The Wikitext-2 contains approximately 2,000,000 words in the training split, and the *Seinfeld* data contains approximately 700,000 words in the training split.

From both data sources the model should gain a better understanding of the English language, specifically regarding sentence structure and syntax. Both text-corpuses are pre-processed similarly to the *Friends* data set.

The advantage of using Wikipedia text is that we have a well structured corpus of text written in grammatically correct English. This should improve our models ability to generate meaningful text. A disadvantage might be that the "tone" of Wikipedia text is too far from that of *Friends*. Wikipedia is written in a very formal way, while *Friends* dialogue naturally is conversational language. This could result in correct but not very "*Friends*-like" dialogue generation.

Trying to test and possibly counter this effect, we also pretrain on a different text corpus with a similar "tone", namely, dialogue from *Seinfeld*. This should also improve generalizability while keeping the conversational nature of the text intact.

5. RESULTS

In this section we show a table of our results from the different models on the performance metric perplexity. Then we visualize learned semantic structures through t-SNE on the embedding weights. Finally we display some generated dialogue that illustrates what the network has learned from *Friends.*

5.1 Model performances

MODEL	HYPERPARAMETERS	PERPLEXITY	DATA
Vanilla model	Batch size: 40 Embedding size: 32 Learning rate: 10^{-2} LSTM size: 8 LSTM layers: 1 Optimizer: AdamW Epochs trained: 100	Validation: 235.8	PTB
Tuned Hyper-parameters	Batch size: 40 Embedding size: 256 Learning rate: 10^{-3} LSTM size: 512 LSTM layers: 2 Optimizer: AdamW Epochs trained: 100	Validation: 108.6	PTB
+ DropConnect	Same as tuned model	Validation: 102.4	PTB
+ DropConnect + Variational sequence length	Same as tuned model	Validation: 96.5	PTB
+ DropConnect + Variational sequence length + Dropout (BEST MODEL)	Same as tuned model	Validation: 92.3	PTB
BEST MODEL	Same as tuned model	Validation: 61.9	Friends
BEST MODEL + pretrained on Wiki	Batch size: 40 Embedding size: 256 Pretrain Learn rate: 10^{-3} Finetune Learn rate: 10^{-4} LSTM size: 512 LSTM layers: 2 Optimizer: AdamW Epochs trained: 100	Validation: 69.1	Friends
BEST MODEL + pretrained on Seinfeld	Batch size: 40 Embedding size: 256 Pretrain Learn rate: 10^{-3} Finetune Learn rate: 10^{-4} LSTM size: 512 LSTM layers: 2 Optimizer: AdamW Epochs trained: 100	**Validation: 60.8** **Test: 60.0**	Friends

Table 1: Different model architectures and performance on PTB and Friends data sets.

Adding complexity to the vanilla model by increasing the sizes of the embedding, number of nodes in the LSTM and the number of layers gives a drop in perplexity from 235.8 to 108.6. Regularization using DropConnect, variational sequence length and Dropout further decreases the perplexity to 92.3.

All pretraining is performed with the same network architecture and settings as found best for the PTB data set, with a learning rate of 0.001 for pretraining, and a fine-tuning learning rate of 0.0001. Using pretrained models on PTB yields no better perplexity results.

When shifting data set from PTB to *Friends* we see a drastic drop in perplexity. This could be due to the fact that word usage is distributed more uniformly across the vocabulary in PTB and the most frequent words are used much more frequently in *Friends* than in PTB. To underline this, a popular fun fact is that the 100 most frequent lemmas account for 50 % of the words used in the Oxford English Corpus (Oxford Dictionaries) [2].

Using the best model architecture found when analyzing PTB data set, resulted in a validation perplexity of 61.9 on the *Friends* data set.

Pretraining with Wikitext-2 causes no increase in performance for the model on the *Friends* data set. It is however noted that the initial loss for the model is lower (A perplexity of 181.3, rather than 244.7), indicating that some information from the Wikitext data set improved early performance.

Pretraining on *Seinfeld* data shows a decrease in perplexity as seen from the perplexity noted in **Table 1**. It was also noted that the initial perplexity was much lower in this case (127.7 compared to 244.7) when compared to no pretraining.

The lowest obtained test perplexity on the *Friends* is 60.0 when using the pretrained model on *Seinfeld*.

Despite *Seinfeld* only accounting for a third of the words presented in the Wikitext, the performance was still increased, indicating that the pretrained text domain is impactful for the specific text classification task.

Further improvements to the model in regards to pretraining can be achieved by increasing the amount of pretraining data, i.e. by utilizing more dialogue similar to that of *Friends*. Previous studies of NLP transfer learning has identified various techniques that can improve performance, such as gradual unfreezing of layers and discriminative learning rates [1]. Inclusion of methods such as these in the *Friends* model could lead to an even lower perplexity.

Performance could further be enhanced by utilizing some of the state-of-the-art techniques for text classification, while trialing deeper networks, as proposed in [4].

5.2 t-SNE on the embedding weights

To further illustrate the specific semantic structures that our network has learned we perform t-SNE on the embedding weights and visualize these in a scatter plot. When assessing the plot we hope to see that words used in similar circumstances, that is that words with some degree of interchangeability are plotted near each other. We assume this is going to be the case for words that are similar in the general usage of English, say words such as “did” & “do” or “so” & “well”, but we also hope to find semantic structures that are specifically related to the Friends universe. We performed t-SNE on the 3500 most frequently used words and manually went through a 2D scatter plot with all these words to find clusters of words that were meaningful to display. The t-SNE scatter plot of 80 words within these found cluster can be seen in the appendix [6].

When assessing the plot we find several interesting clusters. One cluster contains everyday-language type of verbs such as “gonna”, “wanna”, “gotta” but also “do” and “did”. Another cluster has words like “how”, “why” and “when”. A third cluster refers to amounts (“one”, “two”, “many”) and a fourth contains time (“weeks”, “hours”, “months”, “days”). All in all, the network definitely has learned clusters of words we humans use in similar or interchangeable ways, for example “I will come back in two *weeks, hours, days, months”*. Finally, we find that the characters are grouped together, both with or without the semicolon at the end (joey:), but that these clusters are far from each other. This shows that the names of the characters have very different functions in the text corpus when used as “joey” and “joey:”. This makes sense, since the former is as a part of a spoken line within some dialogue, and the latter is a way of denoting who is saying what.

Furthermore, we found an interesting example that illustrates the multiple underlying meanings word clusters can have. This can be seen when looking at a subplot of the t-SNE visualization, here:

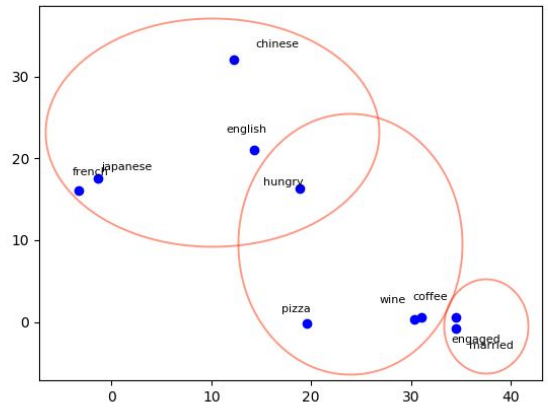

Figure 1: t-SNE subplot on Friends embedded words.

Here we see that nationalities are clustered together and food/drink items are clustered together. But we also see that these clusters are quite close to each other. This is possibly the case, since *chinese* can be both the nationality chinese and chinese food. This is not uniquely related to *Friends,* but the gang definitely talk a lot about food and eat a lot of fast food. So we are satisfied with this result.

5.3 Generated dialogue

Here we show a subset of the generated dialogue that illustrates what the network has learned. The dialogue is generated using our dialogue generator, which can be accessed and tried through our github repository [5].

```
[ scene: rachel and ross enter with the bathroom , ross enters . and ross are there to find her a cup , and the gang who has to come in . ]

phoebe: oh , hey - ho , hey rach ?

ross: i just wanted her .

rachel: i was thinking of the most of those guys were having the last day .

ross: i don't know !

ross: yeah ?

monica: i think i'm going out of this .
phoebe: i am . ( to ross and ross are shocked , and rachel and phoebe )

monica: what ?

ross: oh - huh , i'm gonna be in here ! ( to ross again ) you can get it out .
```

As seen from the generated dialogue, the network is quite good at managing the syntactic structural part of sentence-generation, but is not quite capable of generating an overarching meaningful conversation just yet. The network is doing well at capturing underlying dialogue structure by alternating between the participants in the conversations, i.e. questions are responded to by someone else than the question asker. It also seems to be modelling the characters going in and out of a room, and remembering this throughout the shown text subset. In the context of manuscript writing it is interesting to see that the model remembers to close an opening bracket and close an opening parenthesis, where the content within these are primarily composed of actions rather than speech.

6. CONCLUSION

In conclusion we find that our model has learned a substantive amount of information from *Friends* and we are satisfied with the effect of the steps we have taken to improve our model along the way. We built a well performing RNN-LSTM model on the PTB-data set using some of the state-of-the-art techniques with a validation perplexity of 92.3. After having achieved a satisfying result on the PTB data set, we shifted our focus to the *Friends*-data set. We sought to improve model performance by pretraining on Wikipedia text and dialogue from *Seinfeld.* Wikipedia did not have the effect we hoped for, but pretraining on *Seinfeld* did improve the model, indicating that the pretrain text domain had an impact on the specific text classification task. When assessing learned model semantics through t-SNE dimension reduction, we found several meaningful clusters. Finally, we are satisfied with the generated dialogue and feel that it is quite sensible - when taking the general state-of-the art into account.

7. REFERENCES

[1] Howard J. and Ruder S. Universal Language Model Fine-tuning for Text Classification. *arXiv:1801.06146v5* , 2018.

[2] Oxford Dictionaries. The OEC: Facts about the language

[3] Wikitext-2 Data Set

[4] Merity S. , Keskar N. and Socher R. Regularizing and Optimizing LSTM Language Models. *arXiv:1708.02182v1* , 2017.

8. APPENDIX

[5] Link to Github Repository

[6] t-SNE plot of embedded words from *Friends*
(We have added the red circles ourselves)