DD2424 Project - Group 12
Text Generation

Tristan PERROT tristanp@kth.se
Adrien JOUANNY jouanny@kth.se
Paul MAUDUIT mauduit@kth.se
Arthur DEPRET depret@kth.se

Abstract

This project concerns the generation of Lewis Carroll-like sentences using different architectures of neural networks (RNN, LSTM, Transformers). Additionally, we evaluate the results both quantitatively and qualitatively, employing various methods to improve these results. The methods used involve optimizing the hyper-parameters of the networks as well as modifying the input data (such as data augmentation and embedding techniques).

1 Introduction

The problem we aim to address is natural language generation, commonly used for text prediction/-completion, content summarization, and chat-bots. A significant challenge in this field is evaluating the generated data, as there are numerous valid outputs and various qualitative and quantitative methods to assess a model and its outputs.

The main objective of this project is to compare different architectures and their effectiveness in generating coherent text. Additionally, we evaluate overall performance metrics, such as inference time and training time. We will optimize the results of the architectures through hyper-parameters tuning and employ data augmentation to achieve better generalization.

2 Related Work

Research in natural language generation has advanced with various neural network architectures [6]. Recurrent Neural Networks (RNNs) have been foundational but suffer from limitations like vanishing gradients. Long Short-Term Memory (LSTM) networks address these issues with memory cells and gating mechanisms, handling long-term dependencies better. More recently, Transformer architectures enable parallel processing of text, leading to superior performance and efficiency in generating coherent sentences [2].

3 Data

3.1 Original Data

The data used in this project is the entire text of **Alice's Adventures in Wonderland** by Lewis Carroll, obtained from the Gutenberg Project. This book contains approximately 26500 words. For our experiments, the text was divided into training, validation, and test sets. The validation test is composed of the second to last chapter and the test set is composed of the last chapter. The training set is the rest of the book.

The preprocessing steps were minimal, focusing primarily on splitting the raw corpus into individual characters, as our initial experiments were conducted at the character level. We created a dictionary to map each character to a unique integer, facilitating the conversion of text into a numerical format

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

suitable for model training. This approach allowed us to capture fine-grained details of the text generation process, though it required careful handling of character sequences to maintain coherence in the generated text.

3.2 Data augmentation

As suggested in the guidance, we implemented data augmentation. We used the articles [4] [1] as a guidance to implement it. There are multiple type of data augmentation, such as: random insertion, random deletion, characters replacement, back translation, noise injection. In our situation we choose synonym replacement. This is relatively basic because it consists of replacing randomly some work in a sentence with a synonym. In the appendix A we can see some example of replacement. We choose to augment the text two times, concretely this means that we increase the number of unique words in the vocabulary from 3181 to 5628, that is a 77% increase. We could have increase more the number of reformulation but it takes a lot more time for a result that is not better. As we will see after, if we increase the number of reformulation we loose the author's ton. Moreover we need to remember that increasing the number of words, increase the time needed for training.

4 Libraries used

We base our work on the **PyTorch** library to efficiently train our models on GPUs, saving time. For text processing, we used the **NLTK** library to retrieve and process the data, a crucial step in training models for text generation.

5 Methods

5.1 Training the networks

Initially, we aimed to begin with the simplest foundation to facilitate comparisons with other architectures. For nearly all the networks, we utilized the Adam optimizer [5] and an embedding layer. The Adam optimizer aids in determining the optimal learning rate throughout the training process, while the embedding layer provides a more effective representation of the inputs in a fixed dimension. For the training, we also used dropout and layer normalization to get a better training result with more generalization.

5.2 Generating sequences

Several methods are available to generate sequences using Recurrent Neural Networks (RNNs) [8] . We experimented with three of them:

Deterministic sampling involves picking the word with the highest probability at each step. This approach produces the most predictable and coherent text, which is especially useful for tasks requiring high reliability. Additionally, it is straightforward to implement and understand. However, it often results in repetitive and monotonous text, as it tends to overuse common phrases and words, leading to less creative outputs.

Temperature sampling scales the probabilities with a temperature factor before sampling from the new distribution. A temperature value below 1 leads to more predictable generations, while a value above 1 results in less predictable, more diverse, but potentially less grammatically correct outputs. This method allows fine-tuning of the randomness in text generation, enabling a balance between coherence and diversity. Its versatility makes it suitable for different types of content, from highly structured to more creative texts. On the downside, finding the optimal temperature can be challenging and may require extensive experimentation. Additionally, high temperatures can lead to nonsensical or grammatically incorrect sentences.

Nucleus sampling (also known as top-p sampling) defines a threshold p and only keeps the smallest set of words whose cumulative probability is above this threshold. Words are then sampled from this truncated distribution. This method strikes a balance between determinism and randomness, often resulting in coherent yet diverse text. By eliminating unlikely and potentially irrelevant words, it improves the overall quality of the text. However, it is more complex to implement compared to

deterministic sampling, and the choice of threshold p can significantly impact the output, requiring careful fine-tuning.

Each method has its own advantages and trade-offs, making them suitable for different types of text generation tasks. Deterministic sampling is ideal for scenarios requiring high reliability and consistency, temperature sampling offers flexibility in creativity, and nucleus sampling provides a good balance between the two.

5.3 Evaluation Metrics

To evaluate the performance of our text generation models, we used several metrics:

Loss: We primarily monitored cross-entropy loss during training, which measures the difference between the predicted and actual word distributions. Lower loss indicates better model predictions.

Type-Token Ratio (TTR): This metric measures lexical diversity by calculating the ratio of unique words (types) to total words (tokens) in the generated text. A higher TTR indicates more diverse vocabulary.

Percentage of Correctly Spelled Words: This metric evaluates the spelling accuracy of the generated text by comparing each word to a standard dictionary, providing a percentage of correctly spelled words.

Perplexity: Perplexity assesses how well the model predicts a word sequence, with lower values indicating better performance. It is the exponentiated average negative log-likelihood of the predicted word sequence.

BLEU Score: The BLEU score [7] evaluates the quality of generated text by comparing it to reference texts, measuring the precision of matching n-grams. Higher BLEU scores indicate better alignment with human-written text.

Using these metrics, we ensured a comprehensive evaluation of both the training process and the quality of the generated text.

6 Experiments

In our experiments, we systematically evaluated various aspects of our text generation models. The primary focus was on comparing different architectures, hyperparameters, and regularization techniques to optimize performance. The experiments were structured as follows:

6.1 Generation Process

As described earlier, there are several methods to generate texts and we compared them.

This study was performed on our best model, consisting of a 2-layer LSTM with a 100-dimensional embedding, a context of 200 characters, 512 hidden nodes per layer, a dropout of 0.4 and layer normalization applied between the LSTM and fully connected layer.

Figure 1 shows the different metrics, compared to a reference text. We see that the deterministic method has the lowest TTR, this is due to the fact that it often falls into a repetition loop. Temperature and nucleus clearly outperforms the deterministic method, and we also see a slight improvement when adding nucleus sampling over the temperature, with perplexity going a bit down, approaching the reference text.

This is why we use the nucleus sampling method in the following experiment to assess the performance of text generation.

6.2 Model Architecture Comparison

We compared the performance of RNN and LSTM networks with one and two layers to evaluate the impact of model complexity and depth on text generation.

LSTM models consistently achieve lower training and validation losses compared to RNNs as can be seen in figure 2, indicating better learning and generalization. The two-layer LSTM converges faster and reaches a lower loss.

In terms of text metrics, in figure 3, RNNs show higher TTR, suggesting more diverse vocabulary usage. However, LSTMs have higher perplexity, indicating less confident predictions. Despite this, LSTMs generate text with higher spelling accuracy and achieve higher BLEU scores, reflecting closer similarity to the reference text. The two-layer LSTM performs the best overall.

In conclusion, while LSTMs exhibit higher perplexity, they outperform RNNs in terms of loss reduction, text coherence, and accuracy. The two-layer LSTM, in particular, demonstrates superior performance, making it the preferred model for text generation tasks in this comparison.

6.3 Hidden Nodes Analysis

We studied the impact of the number of hidden nodes in the network to balance model complexity and computational efficiency.

Figure 4 shows that increasing the number of hidden nodes generally improves validation loss but also increases the risk of overfitting, as indicated by the rise in validation loss after several epochs. To mitigate this, regularization techniques are necessary. Additionally, training larger models is computationally intensive, necessitating careful consideration of resource constraints.

In terms of text metrics, we see in figure 5 that the TTR score increases with more hidden nodes, indicating more diverse vocabulary usage. Perplexity decreases with more hidden nodes, showing improved model confidence and coherence. Spelling accuracy is highest for the 128-node model, suggesting better vocabulary learning. The BLEU score is relatively high across all models, with smaller models achieving the highest scores.

In conclusion, while increasing hidden nodes can enhance performance, it requires regularization to prevent overfitting and involves higher computational costs. An optimal range of hidden nodes, particularly around 256, appears to balance these factors well.

6.4 Hyperparameter Tuning

We explored how batch size affects the learning process. Batch size determines the number of samples processed before updating the model parameters.

As shown in Figure 6, increasing the batch size speeds up the training process (in terms of update steps); however, the validation loss tends to increase with larger batch sizes. Training with a lower batch size results in slower training but generally leads to better validation performance.

In terms of text generation metrics, we observe in figure 7, a batch size of 256 achieved the highest TTR score, indicating more diverse vocabulary usage. Despite having the highest perplexity, this batch size also attained the highest BLEU score, suggesting that it generated text most similar to the reference. The BLEU score and spelling accuracy did not vary significantly with batch size.

In conclusion, while larger batch sizes accelerate training, they may lead to higher validation loss. A batch size of 256 strikes a balance, achieving high diversity and similarity to the reference text.

6.5 Regularization Technique

To address overfitting, we experimented with dropout, a regularization technique that randomly sets a fraction of input units to zero at each update during training. This helps prevent the network from becoming too specialized to the training data.

As shown in Figure 8, applying dropout increases the training loss slightly but significantly reduces the validation loss and prevents it from rising too quickly.

In terms of text generation metrics, higher dropout values lead to lower TTR scores according to 9, indicating reduced diversity in the generated text. However, dropout improves the percentage of correctly spelled words and slightly increases the BLEU score.

Thus, it is important to find a balance in dropout. For our best model in Experiment 1 (6.1), we selected a dropout value of 0.3 to balance these factors effectively.

6.6 Number of layers

We investigated whether increasing network depth improves performance by training 2, 3, and 4 layer networks.

In terms of validation loss (figure 10), deeper networks resulted in higher validation losses, indicating increased overfitting with more layers.

However, the text generation metrics present a mixed picture as can be seen in figure 11. Deeper networks achieved higher BLEU scores and slightly better spelling accuracy, but had lower TTR scores, indicating reduced vocabulary diversity.

While adding regularization techniques might improve the performance of deeper networks, this is beyond the scope of this study.

6.7 Transformer Network

We developed a transformer-based architecture [9] comprising an encoder and decoder, utilizing multihead attention mechanisms. The resulting model consists of 2.5 million trainable parameters. In this implementation, we used words as tokens for the input rather than individual characters. This approach proved to be one of our most effective architectures for text generation.

Even though the results are very good for a human, according to our metrics (figure 12)

6.8 Data Augmentation

Finally, we explored data augmentation methods to increase the diversity of our training data. This involved generating additional training samples by applying various transformations to the original data, which aimed to improve the model's robustness and its ability to generalize to unseen data.

We trained a network with the same parameters as our best model on an augmented dataset.

While the training loss was slightly higher, we observed benefits in text generation. As shown in Figure 13, the Type-Token Ratio (TTR) score decreased, indicating reduced diversity. However, we achieved higher spelling accuracy and an improved BLEU score.

The higher loss can be attributed to the use of augmented data, where words with similar meanings are used but may not align with the author's original style. This leads to more general accuracy but less fidelity to the author's writing style.

To further improve results with data augmentation, implementing contextualized word embeddings could be beneficial.[3]

7 Results

The firsts model were really not interesting and we obtain almost random generation (however, some words were understandable even though that was a simple RNN based on char representation) like that:

```
Alice you, yot e, and the n all, S rs. Houcl, int you, ife
    of thilled them!ns lillillis
mbis lyou, 'Why en and ang,' and Alich yealligi, th af
mers the med Alichem I tou Hic you armnerie youglaidow
    yriericemealongNlm every wille, you, and the hrick, you,
    allown and youadveen ililllembell fong th you, ifutilleem ed
    solllow!' mear ther shng Mer and QWr ace,' th?lllairie
'it,it,' Ilar ter her wou, inl,' h
mem doingn it thise you, thing I rig
 er ma, as wast ulilled nchilled headved Iled Alring!' A
```

With our best LSTM model we obtained:

```
Alice was a little house, and said 'That's very curious.'
'I won't interrupt at this morning, I suppose?' said Alice.
'It doesn't matter which way you go,' said the Mock Turtle a
    little three of the jurymen.
'It isn't a fine doesn't matter much,' thought Alice, 'to be a
    serpents to go by this before, never!'
The Mock Turtle sighed deeply, and the long grass rustled at her
    arm, and walked off.
'They lived on saying "Come up, and make your face, and the poor
    little thing!' said the Queen.
'It isn't mad.'
'It's the jury-box,' said Alice.
```

And at the very end, the transformer based model output that:

```
Alice was a good deal worse off than before, as the March Hare had
     just upset the milk-jug into his plate. Alice did not wish to
     offend the Dormouse again, so she began very cautiously: 'But
     I don't understand. Where did they draw the treacle from?' '
    You can draw water out of a water-well,' said the Hatter; 'so
    I should think you could draw treacle out of a treacle-well--
    eh, stupid?' 'But they were IN the well,' Alice said to the
    Dormouse, not choosing to notice this last remark. 'Of course
    they were', said the Dormouse; '--well in.' This answer so
    confused poor Alice,
```

8 Conclusion

In this paper, we implemented and compared multiple architectures. We evaluated their performance by analyzing loss, training time, and various metrics designed to assess the quality and accuracy of the generated text. Additionally, we applied data augmentation techniques to enhance the generalizability of our results. Ultimately, our findings indicate that both a well-optimized LSTM and a transformer model yield exceptional outcomes, closely resembling human-generated text.

References

[1] Data augmentation in nlp, April 2019. URL https://towardsdatascience.com/data-augmentation-in-nlp-2801a34dfc28. [Online; accessed 13-May-2024].

[2] Rnn cells: analyzing gru equations vs lstm, and when to choose rnn over transformers, September 2020. URL https://towardsdatascience.com/rnn-cells-analyzing-gru-equations-vs-lstm-and-when-to-choose-rnn-over-transformers-a28e46bee [Online; accessed 20-May-2024].

[3] Bert, elmo, & gpt-2: How contextual are contextualized word representations?, March 2020. URL https://ai.stanford.edu/blog/contextual/. [Online; accessed 13-May-2024].

[4] Data augmentation in nlp: Best practices from a kaggle master, September 2023. URL https://neptune.ai/blog/data-augmentation-nlp. [Online; accessed 13-May-2024].

[5] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[6] Z. K. S. M. D. A. s. Noureen Fatima, Ali Shariq Imran. A systematic literature review on text generation using deep neural network models, May 2022. URL https://www.diva-portal.org/smash/get/diva2:1660383/FULLTEXT01.pdf.

[7] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

[8] R. M. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview, 2019.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

Appendix: Experiment 1 Text generation methods

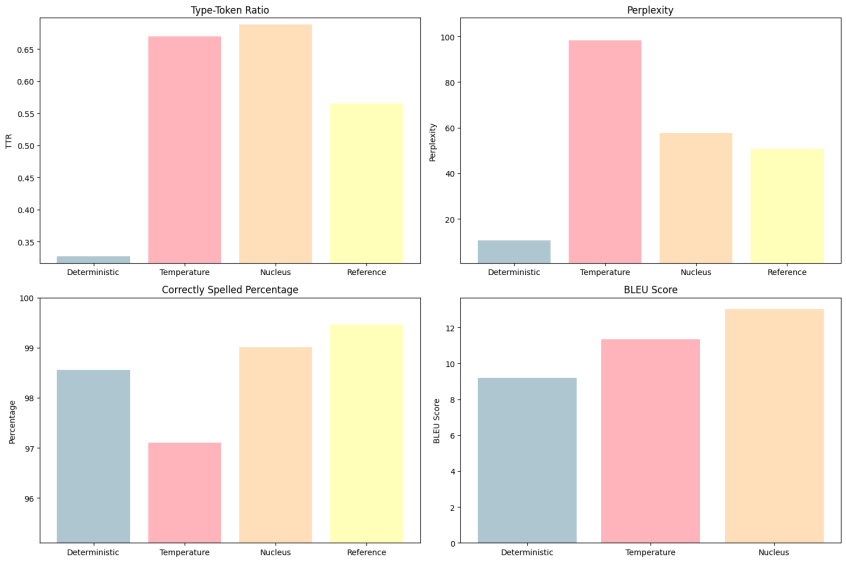

Figure 1: Comparison of text generation methods

Appendix: Experiment 2 RNN vs LSTM

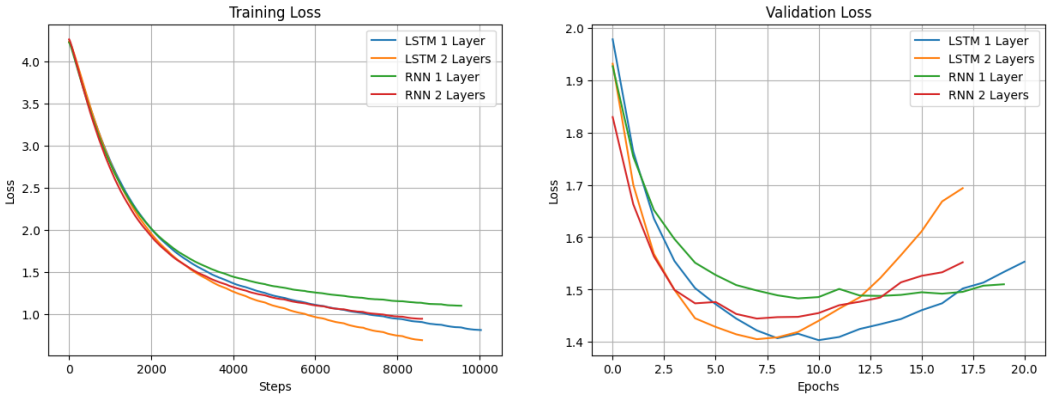

Figure 2: Losses of RNN and LSTM with 1 or 2 hidden layers

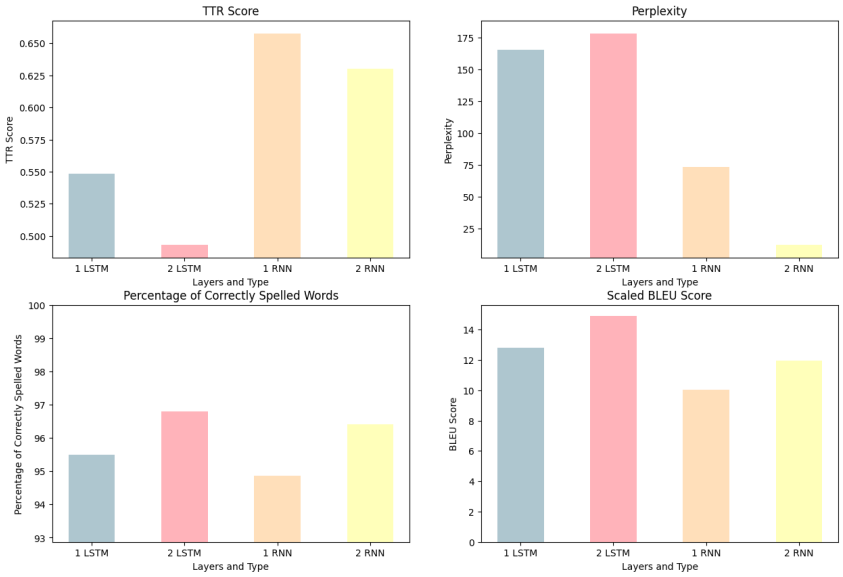

Figure 3: RNN vs LSTM with 1 or 2 hidden layers

Appendix: Experiment 3 Number of hidden nodes

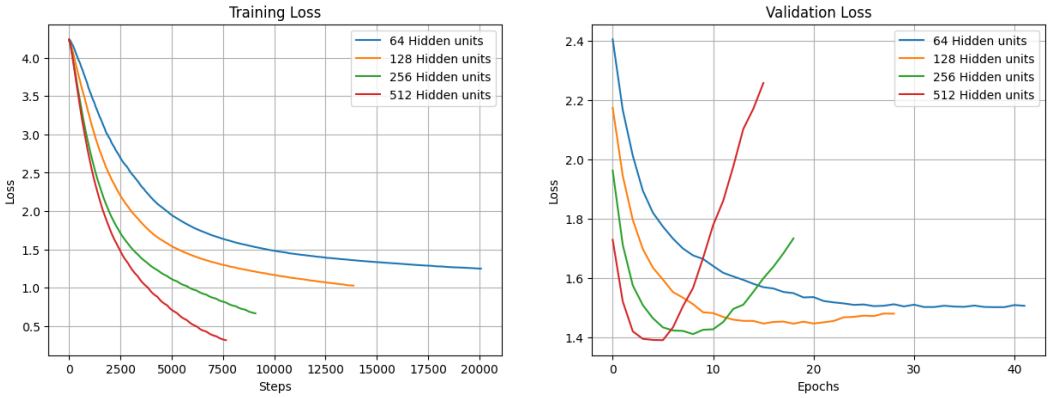

Figure 4: Losses of LSTM when varying hidden size

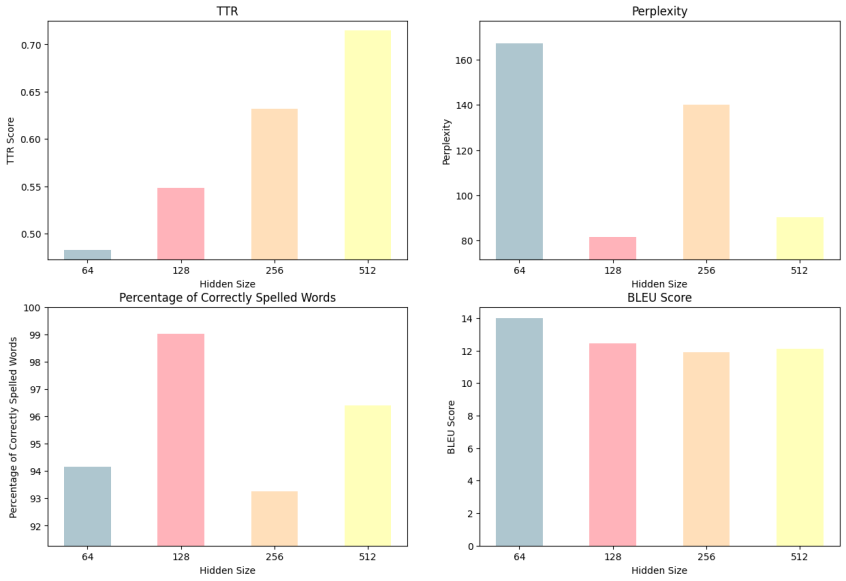

Figure 5: Influence of hidden size on text generation

Appendix: Experiment 4 Batch Size influence

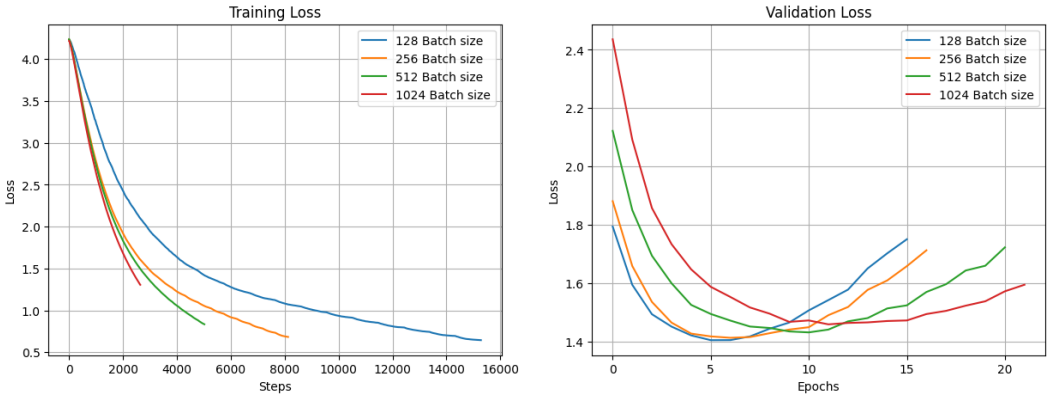

Figure 6: Losses of LSTM when varying batch size

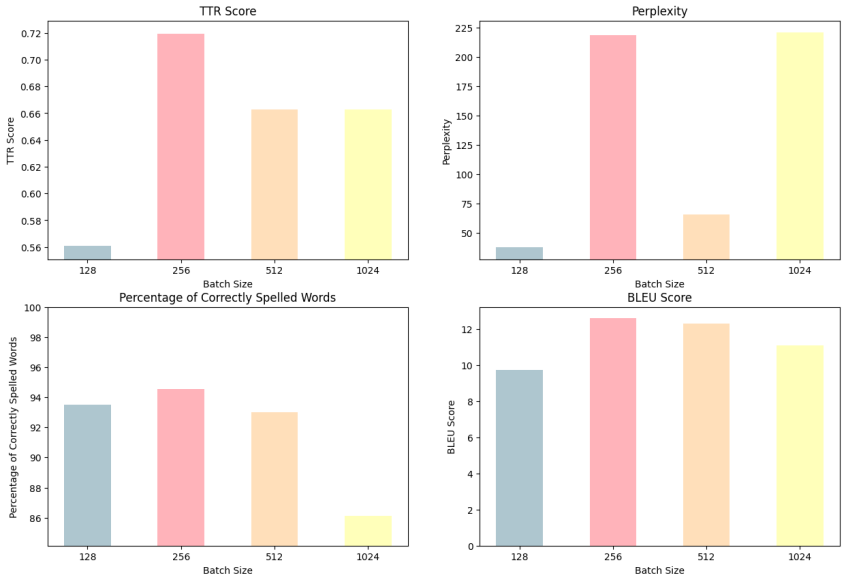

Figure 7: Influence of batch size on text generation

Appendix: Experiment 5 Regularization with Dropout

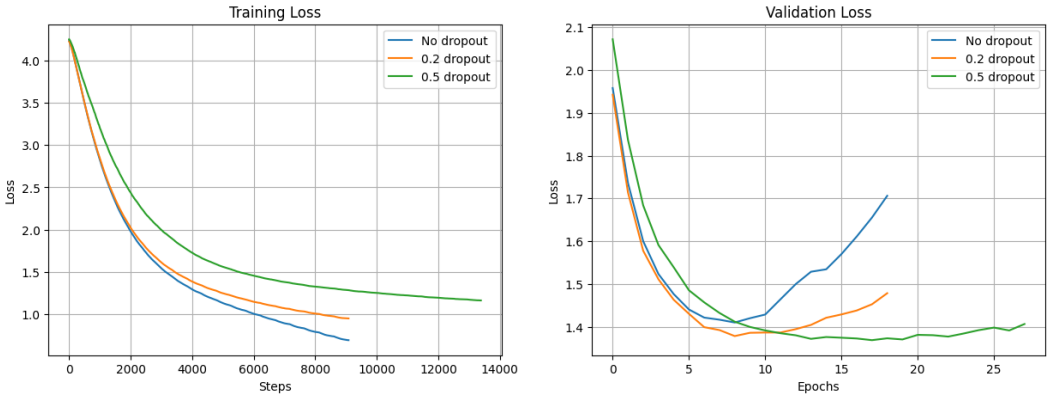

Figure 8: Losses of LSTM when varying the dropout rate

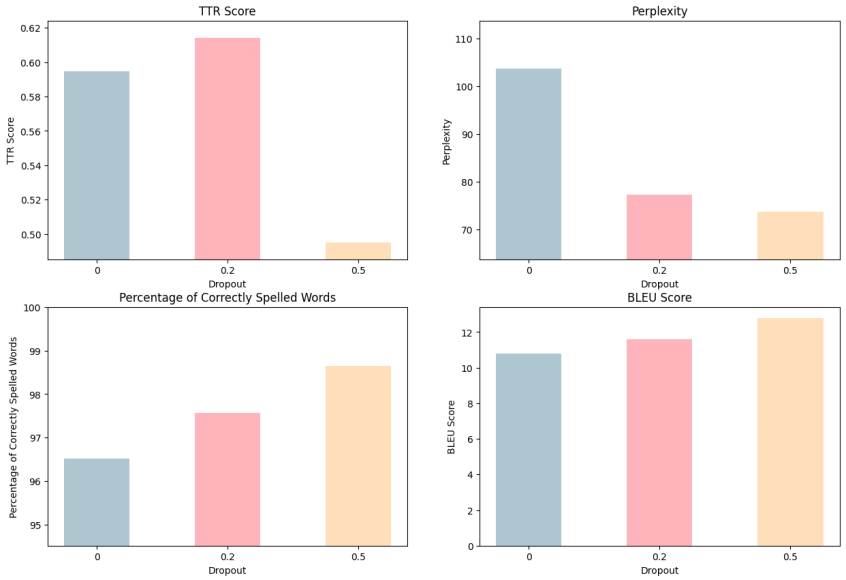

Figure 9: Influence of drop out rate on text generation

Appendix: Experiment 6 Number of Layers

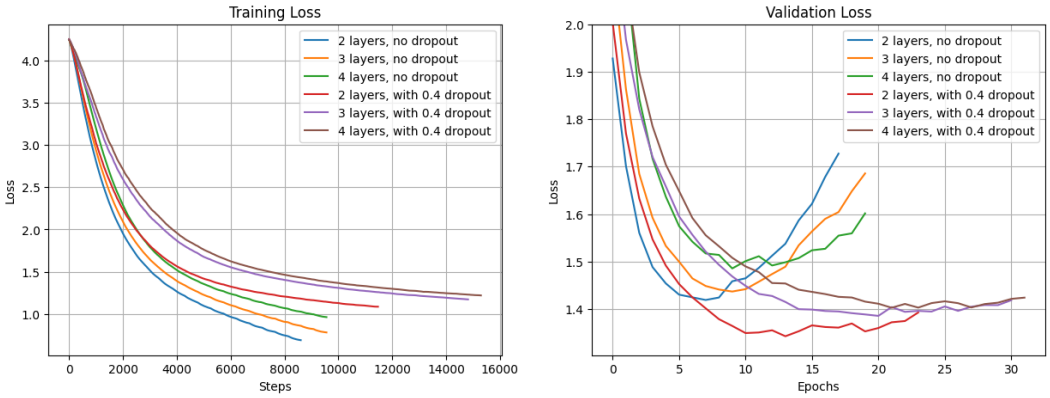

Figure 10: Losses of LSTM when varying the number of hidden layers

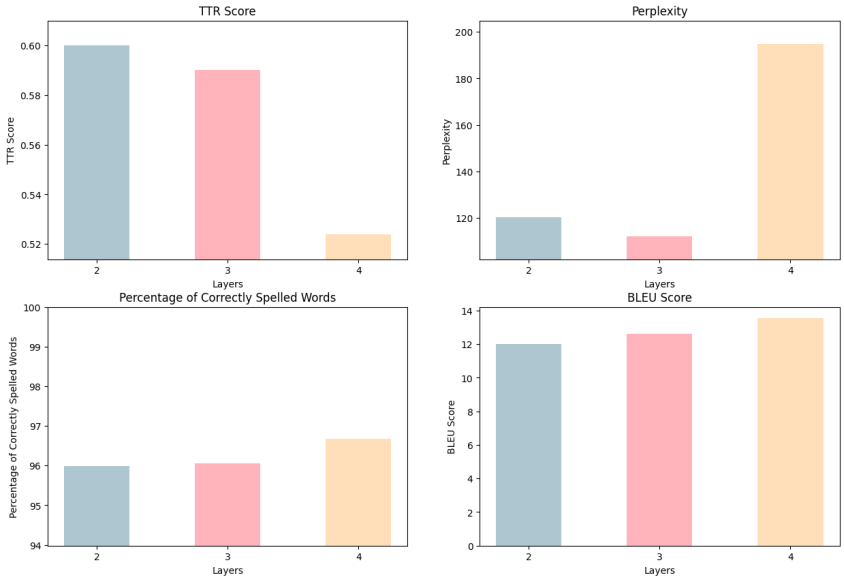

Figure 11: Influence of the number of hidden layers on text generation

Appendix: Experiment 7 Transformer Network

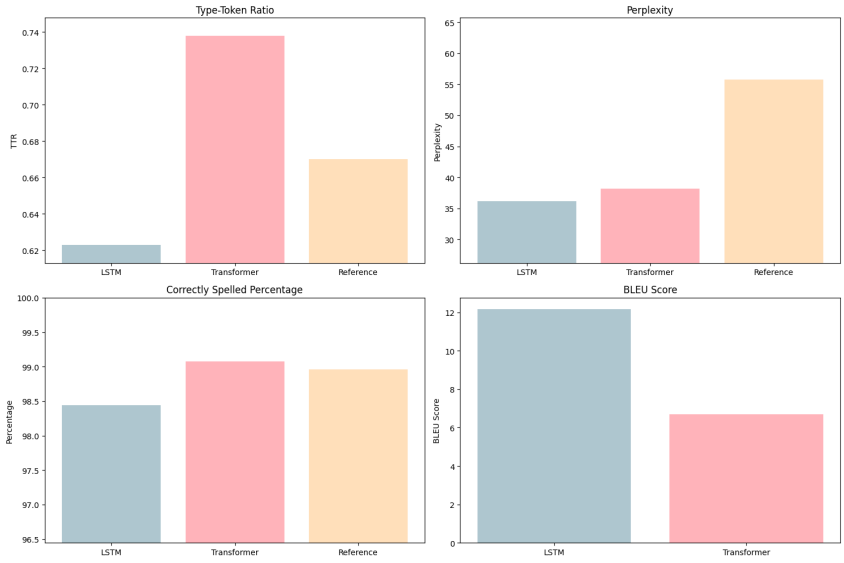

Figure 12: Comparison of text generation: LSTM vs Transformer

Appendix: Experiment 8 Data Augmentation

Here is a short example of the substitution that have been made to augment the dataset. Example 1 of substitution:

- Original: for the hot day made her feel very sleepy and stupid
- Reformulated once: for the hot the made her feel not sleepy years stupid
- Reformulated twice: for when top day made her feel very sleepy and dumb

Example 2 of substitution:

- Original: and looked at it
- Reformulated once: and looked at being
- Reformulated twice: more looked at it

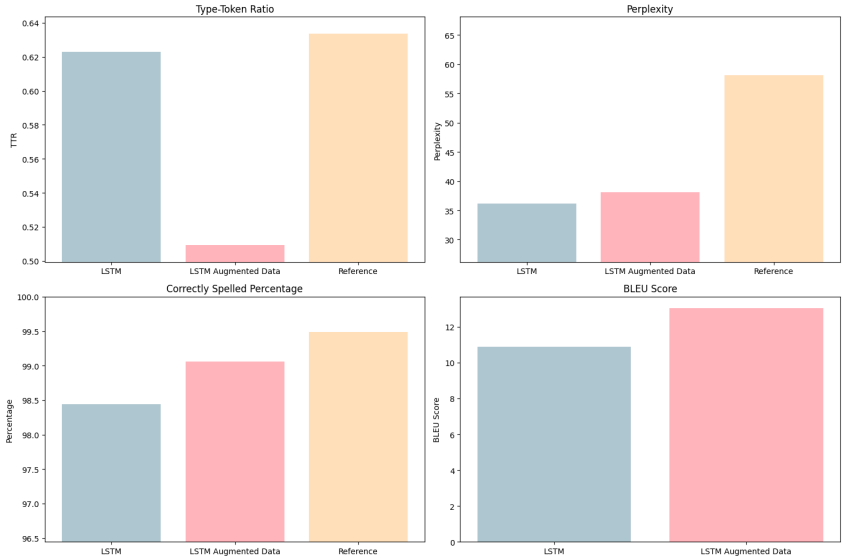

Figure 13: Comparison of text generation: LSTM vs LSTM with Data Augmentation

And here is what a text generation looks like:

alice was all three sides of the house of her and the three gardeners who was the caterpillar took the cook to the thing before she had never doesnt think to alice some called with cat in the way to the baby with a sigh his took nearer in the distance and the party were off if something came with one paw to the mock turtle in the distance but it made me to fancy what the gryphon said was shouted out that she stood well one of the end of its shoes of mouse in silence was a rabbit was over in more little shriek and stupidly until she could not think of an office of the fire and the first same side of the trees had a sort of all the caterpillar and the rabbit was so she did so she began in the air – – what time it to be tried at the confusion that she set off at the mouse began went on of what dark said the pigeon but she sat down again so in the sea so she went on looking at the baby with one eyes and the things growing and said to herself but alice tried to stay down here which he was so suddenly