Comparing RNN, LSTM, and Transformer Architectures for Generation of Harry Potter Text

Mattias Kvist Daniel Ruijs
KTH Royal Institute of Technology
Stockholm, Sweden
{makv, ruijs}@kth.se

Abstract

This project investigates generation of Harry Potter text by comparing three neural network architectures: a vanilla recurrent neural network (RNN), long short-term memory networks (LSTM), and transformer models. Using all seven Harry Potter books, with 6.3M characters and a 107-character vocabulary, we conduct extensive grid searches over hyperparameters and evaluate three tokenization methods, raw characters, Byte-Pair Encoding (BPE), and Word2Vec embeddings. The LSTM outperforms the RNN baseline, achieving better loss and n-gram metrics. The transformers match the LSTMs results with performance scaling with the number of layers, which is not the case for the LSTM. Word2Vec embeddings improve performance slightly, but using Byte-Pair Encoding yields the best results. This suggests that the models are too simple to learn the large number of tokens used with Word2Vec. Future work should explore deeper networks and pre-training on a larger English corpus before finetuning on the Harry potter text. Source code is available at `https://github.com/danielruijs/dd2424-project`.

1. Introduction

This project explores the problem of character-level text generation using deep learning architectures. Given the recent advancements in the field of natural language processing and specifically text generation, it is interesting to understand the strengths and limitations of the most popular architectures. We compare three such architectures: a vanilla Recurrent Neural Network (RNN), a deeper Long Short-Term Memory (LSTM) network, and a transformer model similar to a scaled-down GPT. A simple one-layer RNN is used as a baseline and it is compared to both a one-layer and two-layer LSTMs to investigate how added depth impacts performance. As an extension both a one- and two-layer transformer network is implemented and trained. The hyperparameters of all models are tuned to explore the effects of different configurations. Additionally, Word2Vec and Byte-Pair Encoding (BPE) tokenization is implemented to explore the effects of using better input tokens. A combination of qualitative and quantitative evaluation is used to assess performance. Overall, the results show that the LSTM works better than the RNN, with the transformer performing on par with the LSTM. Increasing the number of layers made no difference for the LSTM, but significantly improved the transformer results. Finally, using better tokenization slightly improved results.

2. Related work

Oleksii Trekhleb's project, ”Shakespeare Text Generation (using RNN LSTM)” [8], demonstrates character-level text generation using a single-layer LSTM trained on Shakespeare's works. Although the results are qualitatively strong and capture the stylistic essence of the source material, the project does not include any comparative analysis with other architectures.

Rami Al-Rfou et al. [1] compare the performance of an LSTM with 2000 units to a 12- and 64-layer transformer model for character-level text generation. Each layer of the transformer models has an embedding dimension of 512 and a dimensionality of the feed-forward network of 2048. The transformer models outperform the LSTM on the text8 dataset [4], with bits per character scores (bpc) of 1.43 for the LSTM, 1.18 for the 12-layer transformer, and 1.13 for the 64-layer transformer. Additionally, the study compares character-level models to word-level models, with the character-level transformer model performing significantly worse than a word-level model.

3. Data

Character-level language modeling aims to generate text one character at a time, capturing fine-grained patterns of syntax and style. The ”Harry Potter Books” dataset from Kaggle [5], curated by Shubham Maindola, provides a comprehensive testing environment and poses challenges for generalization beyond local dependencies. It consists of seven plain text files corresponding to each book in the Harry Potter series. The text files are concatenated into a single string of text and split into individual characters, yielding a total of 6.285.444 tokens with 107 unique characters. The text is divided into training, validation and test sets containing the first 80%, middle 10% and last 10% of the text respectively. This preserves narrative continuity in the validation and test sets, though future work could use sequences from different parts of the text, providing a better measure of generalization. Finally, the text is split into sequences and combined with the target sequences. The sequences in the training dataset are randomly shuffled.

4. Methods

To solve the problem of character-level text generation, three neural network architectures were implemented: an RNN, an LSTM, and a transformer model. The networks were trained to predict the next character in the sequences in the training dataset described above. The validation dataset was used for evaluation during training and hyperparameter tuning and the test dataset was used for final evaluation.

A vanilla RNN was trained as a baseline, followed by a one and two-layer LSTM. Both models were implemented with inspiration from [7]. Using these models, a grid search was performed to investigate the effect of parameters such as batch size, learning rate and the number of hidden units. Table 1 summarizes the tested values. Although a random search could have been used instead, a grid search was chosen as it provides more control over the tested parameters, allowing for a clearer understanding of each parameter's impact.

Table 1. Hyperparameter grid search configuration for the RNN, one- and two-layer LSTM.

Hyperparameter	**Values Tested**
Batch Size	32, 64, 128
Learning Rate	0.0001, 0.001, 0.01
Hidden Units	256, 512, 768, 1024
Sequence length	100

Next, the transformer model was implemented with inspiration from [6] and another grid search was performed with different sequence lengths, number of layers, number of heads, and the dimensionality of the feed-forward and embedding layers. The explored parameter values can be seen in table 2. The best learning rate and batch size from the first grid search were used. All models are implemented using TensorFlow and the Keras API, which facilitates efficient and consistent training processes across architectures, and training is performed on an Nvidia T4 GPU.

Table 2. Hyperparameter grid search configuration for the transformer model. The dimensionality of the feed-forward network was set to $4 \times$ Embedding layer dimension.

Hyperparameter	**Values Tested**
Number of layers	1, 2
Number of attention heads	4, 8, 12
Embedding layer dimension	256, 512
Sequence length	100, 200, 300

4.1. Tokenization

Initially, all models were trained on raw characters with each of the 107 unique characters seen as a separate token. To explore the effect of more meaningful tokenization and richer input representations, two additional techniques were implemented: Word2Vec and Byte-Pair Encoding (BPE). Other tokenization strategies such as Glove were considered but not implemented due to time constraints. For the Word2Vec model, the text was divided into 27313 unique words and trained to generate embeddings. Skipgrams were created from all seven texts with equally many positive and negative samples. A higher ratio of negative samples could benefit performance, but this could not be explored due to hardware limitations. The embeddings were then used to train all models. Byte-Pair Encoding was implemented by extending the base 107 character vocabulary with 1000 additional tokens that represent the most frequently occurring pairs of tokens in the text.

4.2. Metrics

In addition to the qualitative evaluation, quantitative evaluation is done using multiple metrics. Firstly, the loss scores on the validation and test datasets are compared. Additionally, a sample of 1000 characters is generated and the proportion of correctly spelled words (1-gram) and the percentage of 2, 3, and 4-grams are used as quantitative metrics. The n-gram scores represent the percentage of word sequences of length n that appear in both the generated text and the training text. Other metrics such as bits per character were considered but not implemented due to limited time. These metrics could have served as a better indication of the performance of the models.

5. Experiments

The first hyperparameter search showed that a learning rate of 0.001 yielded the best validation loss and n-gram scores. Further, the greater number of hidden units yielded better results, while the batch size made minimal difference. Using these results, the final RNN and LSTM models were trained with a learning rate of 0.001, 1024 hidden units, a sequence length of 100, and a batch size of 64.

The grid search for the transformer model showed that the greater number of heads and layers improved performance as expected. Increasing the sequence length also slightly improved performance. Interestingly, a lower dimensionality of the feed-forward and embedding layers yielded better results. For this reason, the final transformer networks were trained with 12 attention heads, an embedding dimension of 256, feed-forward layer dimension of $4 \times 256 = 1024$, and a sequence length of 300.

Table 3 summarizes the results for the RNN, LSTM and transformer models. The LSTM models clearly yield better performance than the RNN, both in terms of loss and n-gram scores, and the 2-layer transformer is on par with the LSTM. Interestingly, the one- and two-layer LSTM models perform almost identically, but for the transformers the additional layer improves performance significantly. This suggests that the transformer model could surpass the LSTM models with enough layers. Additionally, transformers are known to require a lot of data to perform well [3], [2], which means that the relatively small dataset used could be a limiting factor. Samples generated by the character-level models can be found in the Appendix. The samples again show that the LSTM and transformer models produce more coherent and realistic text compared to the text produced by the RNN.

Table 3. Model performance comparison. The table shows test loss and n-gram scores in percent for each architecture. LSTM-{1,2} and Transformer-{1,2} refer to the one- and two-layer variants, respectively.

Model	Test Loss	1-gram	2-gram	3-gram	4-gram
RNN	1.22	93.0	60.0	17.2	1.8
LSTM-1	1.15	92.4	60.2	20.0	4.7
LSTM-2	1.15	91.8	64.7	20.1	4.2
Transformer-1	1.29	85.5	47.2	13.6	2.3
Transformer-2	1.18	91.8	58.8	20.1	6.0

5.1. Tokenization

To test the generated embeddings from the Word2Vec model, a few experiments were conducted. First, the words closest to ”Harry” and ”magic” in the embedding space were found to be ”Potter” and ”spells”, respectively, using cosine similarity. Next, addition of words was done in the embedding space and the closest words to the sum were computed using cosine similarity. Adding ”Harry” and ”owl” returned ”letter” and ”Hedwig” as the closest words, and adding ”Harry” and ”sock” yielded ”Dobby” as the second-closest word. Finally, multiple items, spells, and character names were selected and visualized in two dimensions using PCA, as can be seen in figure 1 of the Appendix. Words from different categories were clearly separated, indicating that the Word2Vec model had learned embeddings that distinguish words. These experiments indicate that the embeddings work well.

Table 4 summarizes the results for all models trained with Word2Vec. The 1-gram scores are almost 100% for all models, which is expected as tokens more closely represent words. The 2-, 3- and 4-gram scores only see slight improvements, with all models performing similarly. This could be due to the very large number of tokens (27313) that the models have difficulty learning. Note that the loss is not comparable to the loss in Table 3 due to the much larger number of tokens.

Table 4. Model performance comparison when using Word2Vec. The table shows test loss and n-gram scores in percent for each architecture. LSTM-{1,2} and Transformer-{1,2} refer to the one- and two-layer variants, respectively.

Model	Test Loss	1-gram	2-gram	3-gram	4-gram
RNN	4.97	99.5	66.8	24.9	4.3
LSTM-1	4.72	100.0	68.7	24.4	4.4
LSTM-2	4.87	100.0	60.5	16.7	1.3
Transformer-1	5.01	99.9	67.1	15.9	1.0
Transformer-2	4.79	99.7	69.0	19.3	2.8

Finally, table 5 summarizes the results for the models trained with Byte-Pair Encoding. As expected, the 1- and 2- gram scores see slight improvements compared to the character-level models. The results are also better than with Word2Vec, which could imply that the models perform better with fewer tokens. It should again be noted that the loss is not comparable to tables 3 and 4 due to a different number of tokens.

Table 5. Model performance comparison when using Byte-Pair Encoding. The table shows test loss and n-gram scores in percent for each architecture. LSTM-{1,2} and Transformer-{1,2} refer to the one- and two-layer variants, respectively.

Model	Test Loss	1-gram	2-gram	3-gram	4-gram
RNN	2.26	93.1	63.7	25.6	8.8
LSTM-1	2.14	96.2	67.4	21.4	3.9
LSTM-2	2.14	94.4	67.6	22.1	4.8
Transformer-1	2.24	90.4	56.1	19.1	3.2
Transformer-2	2.21	94.1	65.6	22.1	3.8

The Appendix shows samples of text generated by the 2-layer transformer model with Word2Vec and Byte-Pair Encoding. The samples show that the Byte-Pair Encoding models are able to capture the structure of the text better than the Word2Vec models, as the tokens are more similar to character-level, capturing finer detail. This yields text that looks closer to the text in the books. The models using Word2Vec still struggle with learning long-range dependencies and producing consistent, fluid text, possibly due to the very large vocabulary of 27313 tokens.

6. Conclusion

Overall, the results show that the LSTM generates better text than the RNN, and that the transformer performs on a par with the LSTM. Increasing the number of layers made no difference for the LSTM, but significantly improved the results of the transformer. For future research it would be interesting to see if a transformer with more layers could surpass the performance of the LSTMs.

Using better tokenization only slightly improved results. This could be due to the models being too simple to learn a larger number of classes or because the dataset is too small to yield good results.

An additional aspect that could be explored is transfer learning, especially for the transformers which are known to benefit from being trained on more data. This could be done by first training on a large English text corpus and then fine-tuning on the Harry Potter dataset. In addition, the dataset should be pre-processed by stripping out chapter titles and other headings to improve model performance.

References

[1] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention, 2018. 1

[2] Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer, 2021. 3

[3] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. 3

[4] Matt Mahoney. Large text compression benchmark, 2009. Accessed: 2025-05-05. 1

[5] Shubham Maindola. Harry potter books. https://www.kaggle.com/datasets/shubhammaindola/harry-potter-books, 2024. Accessed: 2025-05-05. 1

[6] TensorFlow. Neural machine translation with a transformer and keras. https://www.tensorflow.org/text/tutorials/transformer, 2024. Accessed: 2025-05-06. 2

[7] TensorFlow. Text generation with an rnn. https://www.tensorflow.org/text/tutorials/text_generation, 2024. Accessed: 2025-05-07. 2

[8] Oleksii Trekhleb. Shakespeare text generation (using rnn lstm). https://github.com/trekhleb/machine-learning-experiments/blob/master/experiments/text_generation_shakespeare_rnn/text_generation_shakespeare_rnn.ipynb, 2020. Accessed: 2025-05-05. 1

A. Additional Results

Sample text produced by the character-level RNN:

```
You know what does! That’s why Hedwig’s direct you."

"The smelled; he had pickuted the Prime

Harry Potter, I separated the leg on painful feather than ever. "You have to occeptaming the Butterbeer... it’s supposed to do so!
Hasting feel the necklace of a window, Harry lay Harry had disappearing onto their wandshoeffer?"
```

Sample text produced by the character-level 2-layer LSTM:

```
At the Ministry of Magic got in a difference as somether stone, but not a frightened from behind you early three silhouted
wizyawars knives as we want to jug prefer up to Divination to brill frog leaving and yet there was everything came enough."

"Nah, listen," said Xenophilius, uppersing that he had not found the connection after every time they didn’t she want.
Sarting was remedied in dust with Luna, Dean Thomas, just behind them and decided to speak to Magical Malvanta! Students.
```

Sample text produced by the character-level 2-layer transformer:

```
Yeah . . .

"Remus . .."

Soon.

They stood, he went looking a loud close to Ron and Hermione, who had seen a chair box and glittering clamber beneath them.
Best’re staying in here, or else that ups to be behind to be visible for the bottom sharp the horror, away straight beside
by the moody black-fireworks.
```

Sample text produced by the 2-layer transformer with Word2Vec:

```
What we saw nearly always getting rid! ’ A thing is, ’ said Moody through the darkness, ’ said Lupin pleasantly. ’ You will do
you are more Headmaster, you Quidditch. ’ said Professor Sprout then? ’ ’ Oh, dear? ’ Yes, and I will leave, Snape extracted
your work. And Avery always my party and it is your shirts buy your dormitory! ’ None of you? Looking as though if it must be
killed sorted with how danger will come to you don Speaking you are no, you do, Potter, Potter, so different action! ’ s enough
to arrange exert ’ In OWL, Mr In conditions started to nine and really, now ... I do I don ’ t ’ This, can, ’ Is it? ’ Professor
McGonagall? Same the usual - ’ said Professor me ... I was just an evening ... And is this time. Professor McGonagall, Potter?
```

Sample text produced by the 2-layer transformer with Byte-Pair Encoding:

```
Now but what happened, the worst new one’s off, repeatoes you keep talking for teres?"

Ron stared at the day about books, Ron and Hermione had dropped it over a parchment to object up on The hospital wing
with the Hermione.

"Yes, Ron, wasn’t you?" asked Ron, apparently punish with him spoke.

And then, Harry drew sevals were echoing to avoid silence in a tunle instead from a final kind of silver potion.

"I know whether You-Know-Who-Who!" said Ron. "You stunged. I don’t feel it’s quite refrightened from Slughorn. Somehow
you can give the House? Didn’t you laughed, who was Petunia? It was the time for ckindly, but a next thing, when I said
you don’t think you fanciating what
```

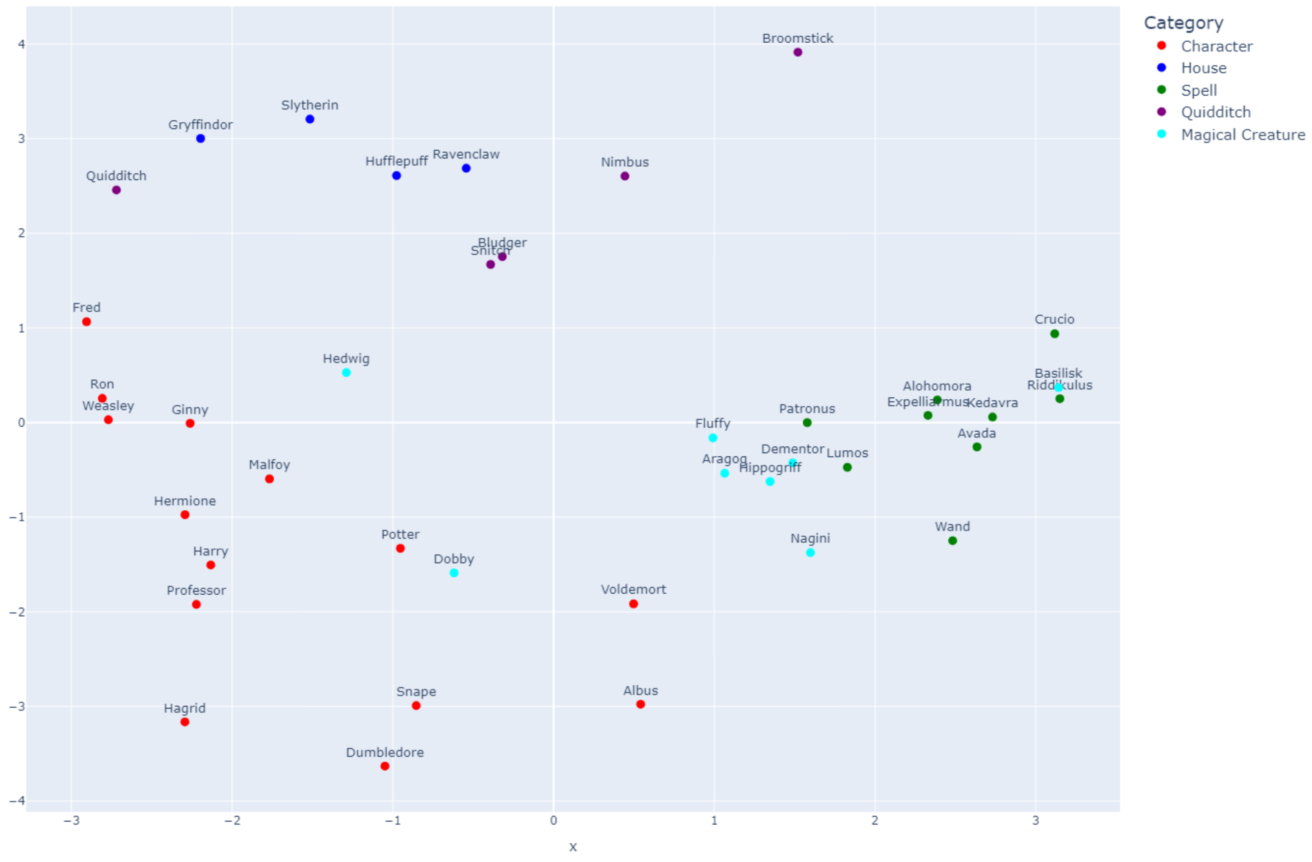

Figure 1. PCA visualization of Word2Vec embeddings.