Shakespearean Text Generation: RNN vs. LSTM Analysis

Nordstrand Mattias mattnor@kth.se
Qorbani Reza qorbani@kth.se
Senane Zineb senane@kth.se

Abstract

In this report, we investigate the effectiveness of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) models in emulating the unique literary style of William Shakespeare. Our models are trained on a corpus of Shakespeare's plays, with the objective of generating text resembling his stylistic nuances. We analyze the performance of different models and their variations, considering factors such as the use of Byte-Pair Encoding (BPE) tokenization, Word2Vec embeddings, and data augmentation techniques. We further conduct a grid search for optimal learning rate and batch size, and examine the influence of varying the number of hidden nodes. Nucleus and Temperature Sampling are employed to balance the novelty and coherence of the generated text. Our experiments reveal intriguing differences in the capabilities of each model, with LSTM demonstrating superior performance in certain scenarios. While data augmentation does not significantly enhance model performance, it has potential in aiding model generalization. The detailed analysis of these findings and the resultant models' text-generation abilities will be elaborated upon in the main body of this report.

1 Introduction

While AI solutions exist for understanding and generating contextually relevant text, their capabilities are tested when replicating the distinct style and complex linguistic structures of Shakespeare's works. Our models emulating this specific style could aid in various fields such as entertainment, presenting fresh and creative ways to engage with Shakespearean content.

We trained Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) models, on Shakespeare's corpus to generate text resembling his unique literary style. We compared quantitatively and qualitatively the models' ability to generate Shakespearean plays. Moreover, we investigated the effect of using word embeddings and Byte-Pair Encoding (BPE) tokenization as well as data augmentation techniques.

The results of our investigation demonstrate intriguing differences in the ability of each model to generate text that aligns with Shakespeare's style, both in terms of quality and stylistic alignment. Detailed exploration of these findings and further implications will be discussed in the Experiment section of this report.

2 Related Work

LSTM [DVS20] and RNN [SMH11] can both be used for text generation. In comparison, RNN can be difficult to train due to exploding or vanishing gradients, while An LSTM, on the other hand, does not have the same problem as RNN for vanishing gradients, and is able to capture long-term dependencies in a way not possible for the RNN.

RNN and LSTM have been compared before when it comes to text generation. For example, in 2019 there was a study [MJM19] comparing LSTM, bidirectional RNN, and Gated Recurrent Unit (GRU) for the generation of scripts (for TV series). Similar to our work, they had an input string as input, which was used as the basis for the script generation, and they experimented with the number of layers.

As in our work, they used temperature for more surprising/unsurprising text generation. In their study, for their measurement of overall performance, they obtained the lowest loss for bidirectional RNN, then LSTM, and then GRU, but the execution time was in the opposite order.

3 Data

The data used in our project comprises a collection of William Shakespeare's plays, sourced from [Sha]. This is a textual dataset, written in Early Modern English, that includes a variety of his works, totaling about 1.115.000 sequences of length 50 in the character level.

To facilitate the generation of augmented plays, we first extracted passages without character names. These passages were then augmented and reassembled to form enhanced versions of the original plays. Additionally, in preparing the data for training the Word2Vec model, we performed another layer of cleaning, where we also retained newline characters as special tokens <NewLine> to ensure their representation in the model's learning process. Afterward, we tokenized the text using two levels approaches. Despite the dataset's availability and Shakespeare's popularity, most resources available are basic TensorFlow tutorials, thus there is a lack of well-defined and state-of-the-art methods for generating Shakespearean text.

4 Methods

4.1 Metrics & evaluation

We made use of Pytorch library for code related to defining and training the models. We tried to make the interface for the RNN and LSTM models as similar as possible, to make it easier during testing. We used Jupyter Notebook to easily train and test them, and used VM on Google Cloud Platform to access GPU. We trained the models with mini-batch gradient descent. We used the Binary Cross-Entropy loss on test data as one metric. Another metric we used was perplexity [Cam22], which is a measurement of how well the model can predict, and can be defined as:

$$\text{Perplexity} = \left(\frac{1}{P(w_1, w_2, \ldots, w_N)} \right)^{\frac{1}{N}}$$

We also looked at the generated texts from a qualitative/subjective perspective.

4.2 Parameter Influence and Grid Search

When we are training our models, we want to select hyperparameters that lead to good performance. Therefore, we investigated the effect of different parameters. More specifically, we investigated we effect of the learning rate, batch size, and the number of hidden nodes. This investigation consisted of two steps.

The first step was a grid search over batch size and learning rate. Grid search means we systematically trained and tested the RNN and LSTM for different parameter combinations, of batch size and learning rate in this case. When we did it, we began by performing a coarse-grained search. This means we selected parameters in a very wide range. Based on the coarse-grained search, we then selected parameters within a more narrow range, which seemed related to good model performance. Thus, we did a grid search to get an indication of what is a good batch size and learning rate.

The second step was investigating the effect of the number of hidden nodes. Therefore, we trained RNN and LSTM models, with different numbers of hidden nodes, and plotted how these led to different perplexity scores.

4.3 Nucleus Sampling & Temperature

In solving the problem of text generation with our Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models, we applied two sampling techniques: Nucleus Sampling (top-p sampling) and Temperature Sampling. Nucleus Sampling was used to address the issue of overly incoherent and repetitive outputs, usually encountered in typical random sampling techniques. By

limiting the sample space to a subset of tokens with the highest probabilities (exceeding a threshold p) , we ensured that our generated text remains coherent and relevant to the context. On the other hand, Temperature Sampling was used to control the randomness of the text generation, hence, acting as a tuning parameter to balance between exploration (diversity) and exploitation (accuracy). Lower temperatures led to less randomness and higher predictability, whereas higher temperatures resulted in more randomness. There are also alternatives such as Top-K sampling (sampling from the k most likely outcomes) and search algorithms such as greedy search, which chooses the word with the highest probability as the next word, and beam search which tries to find the highest probability sequences. Throughout the process, we applied skills and concepts acquired during the course, notably understanding the mechanics of RNNs, LSTMs, and various sampling techniques. The combined approach of Nucleus and Temperature Sampling ensured a balance between generating creative, novel sentences and maintaining coherence.

4.4 BPE and Word Embeddings

In addressing the challenges outlined in the introduction, we employed two different approaches for preprocessing our corpus: Byte-Pair Encoding (BPE) tokenization and Word2Vec embeddings. BPE was chosen as it effectively manages the issue of out-of-vocabulary words, by splitting unknown words into subword units that are likely to appear in the corpus, enhancing the model's capability to handle and generate novel words. Word2Vec, on the other hand, was used to transform our tokenized corpus into vector representations, capturing the semantic similarities between words. This allows our model to generate coherent and contextually appropriate text. While other tokenization methods such as SentencePiece and WordPiece were considered, we focused on BPE as this was mentioned in the project description and the long training time did not allow for extensive experimentation. We also contemplated other word embedding techniques like GloVe, but Word2Vec was used because of the aforementioned reasons. Through the application of BPE and Word2Vec and concepts ingrained during the course, we have ensured that our RNN and LSTM models have a well-prepared, meaningful input representation of the text, optimizing their learning and subsequent text generation abilities.

4.5 Data Augmentation

Data augmentation consists of introducing minor changes to the data while preserving the essential characteristics and features to provide some variations and diversity that the model can learn from and help prevent overfitting. It is less established for textual data, however, in the case of text generation using RNNs and LSTMs, it can be particularly beneficial to improve the models' performance, especially when dealing with limited training data. Textual data augmentation techniques can be categorized into three different sections: character level; where methods perform random insertion, deletion, or swap of characters, word level; involving synonym replacement, and similar approaches as in character level, and finally sentence level, where the context is preserved but represented using different language units. We implemented three different ways of data augmentation using the NLPAug library, The first augmenter substitutes a word by its synonym using wordnets synsets. The second augmenter substitutes the words with the most similar words using pre-trained word2vec models. Finally, the third augmenter replaces words while preserving the same context using a pre-trained Bert. In our experiments, we applied the last type of augmentation, contextual augmentation, as it utilizes the power of bidirectional representations of transformers, and substitutes words but keeps in view the context of the entire sentence before setting this replacement. The data augmentation has been applied to the first 6000 passages of Shakespeare's Plays, 80% of the dataset corresponding to the training set, and the augmented passages have been concatenated back together by adding the corresponding character to get more accurate augmented plays. The validation and test sets remained untouched to check whether the models can generalize well on unseen data and reflect the model performance on new sequences. When enabling augmentation in our experiments we add the first augmented 500000 characters to the original plays.

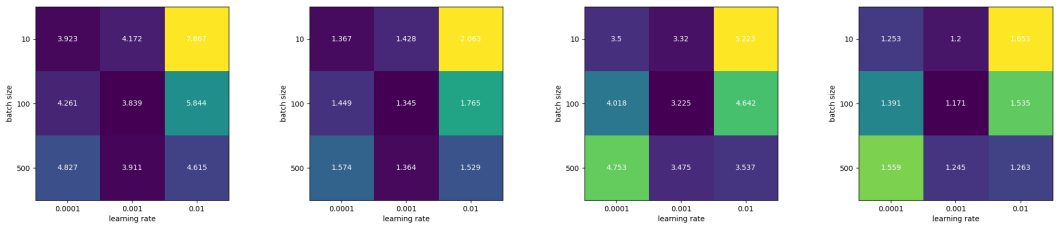
5 Experiments

Our PyTorch-based modeling process, we used the embedding layer for input handling, architecture-specific layers, and dropout for regularization. Each model concluded with a linear layer to ensure the output matched the required size. We examined and compared several RNN and LSTM architectures,

as outlined in Table 1. Our implementation [NQS23] was designed for easy modification of network attributes such as the number of dropout layers, bidirectionality, and the operating level (character or word). The baseline models can, by parameter selection, be more advanced.

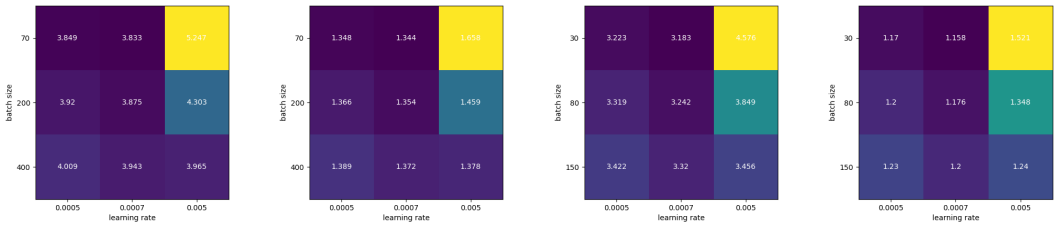
5.1 Hyper parameters tuning

We performed a model parameter investigation by conducting both coarse-grained and fine-grained grid searches. The coarse-grained search showed that for both the RNN and LSTM models, a batch size of around 100 and a learning rate of approximately 0.001 yielded good performance. The fine-grained search further refined these parameter ranges, suggesting a learning rate of 0.0007 and a batch size of 70 with for the RNN model and a batch size of 30 for the LSTM model, see Fig. 1 and 6.



(a) RNN Perplexity (b) RNN Loss (c) LSTM Perplexity (d) LSTM Loss

Figure 1: Coarse-grained grid search over learning rate and batch size, for RNN (A, B) and LSTM (C,D).



(a) RNN Perplexity (b) RNN Loss (c) LSTM Perplexity (d) LSTM Loss

Figure 2: Fine-grained grid search over learning rate and batch size, for RNN (A, B) and LSTM (C,D).

A sufficiently high number of hidden nodes was crucial for the performance (perplexity score) of RNN and LSTM. However, increasing the number of hidden nodes too mycworsens the performance, see Fig. 3

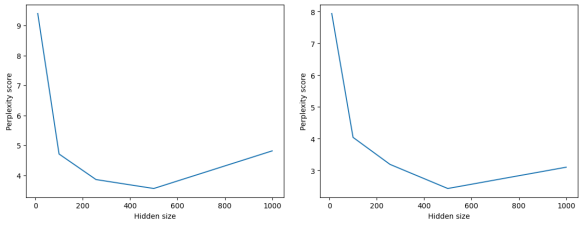
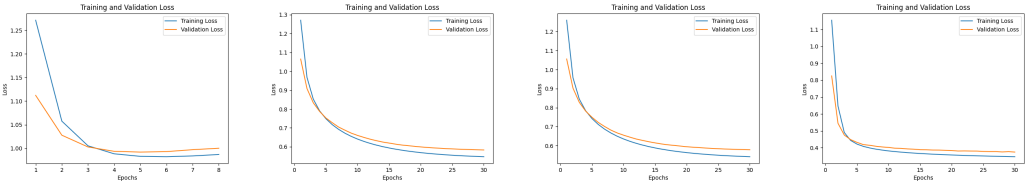


Figure 3: Effect of number of hidden nodes on model performance, for left) RNN, right) LSTM

5.2 Baseline model

We investigated the performance of different architectures, including a character-level RNN as our baseline, a single-layer LSTM, a single-layer LSTM with dropout, and a two-layer LSTM all operating on a character level. The best results were achieved by our two-layer LSTM with a performance of a test loss of 0.374 and perplexity of 1.454, confirming the increased effectiveness of the deeper architecture. Qualitatively, the baseline RNN generated mostly incoherent text, while single-layer LSTM improved syntax but was occasionally irregular. Adding dropout to the one-layer LSTM further refined the text, and enhanced the generalization, by grasping character interactions

and context. The two-layer LSTM outperformed all the aforementioned architectures, generating the most coherent and contextually relevant text, see the appendix 6 for examples. Despite the improved performance with deeper architectures, the character-level models still produced irrelevant words and phrases.

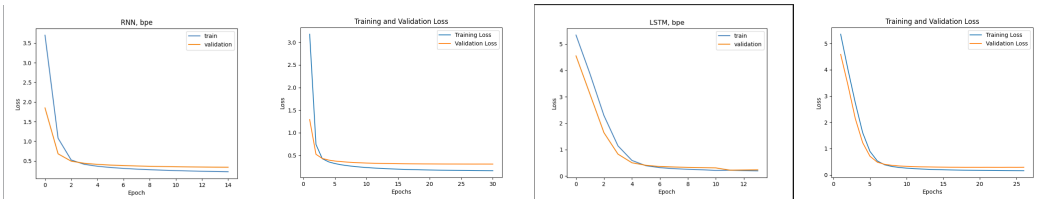


(a) RNN - Baseline (b) 1-layer LSTM (c) 1-layer LSTM+Dropout (d) 2-layers LSTM

Figure 4: Loss plots for Basline and LSTM models trained with character sequences.

5.3 BPE & W2V

In order to assess whether using word-level and BPE tokenization resulted in any improvement or not, we use the same hyper-parameters as the experiments in the word-level. Starting with BPE tokenization, we used BPE implementation in HuggingFace Tokenizers library [Inc19] which is optimized and helps us get an accurate picture of the effect BPE tokenization. The BPETokenizer takes two arguments, vocabulary size with the default value of 30000, and the minimum frequency of occurrence that a token must have to be considered, with a default value of 2. We used the default values for these arguments in the experiments. We applied the tokenizer on both of the baseline models and plotted the train/val losses for these trainings in Figure 5. There was a small issue towards the end of the training for LSTM with BPE tokenization where the environment shut down and we had to load up the model again and we missed passing one iteration when plotting. This led to the sudden decrease towards the end of 5 C, but from the print outputs, we can say that the plot should continue flat from the 11th epoch just as in other plots. From the results in Table 1 we can see that the test loss for the LSTM architecture is 30% lower, which is in line with the assumption that LSTM improves the performance as sequences grow in size. In our experiment we used the Word2Vec implementation by Gensim [ŘS10] which we trained on our corpus. We see the results for W2V applied to both the RNN and LSTM model in Figure 5 and in Table 1. Just as in the experiments with BPE tokenizer, we get a considerably lower test loss with LSTM architecture, indicating once again that LTSM does in fact improve performance. Comparing the text generated by BPE vs W2V, we can say that BPE does, in fact, generate better text, most of the time even memorizing the text completely. BPE can generate a long coherent sequence of text where even characters (in the play) included are relevant (for example a dialogue between Romeo and Juliet when the input is simply ROMEO:).



(a) RNN with BPE (b) RNN Word Level (c) LSTM with BPE (d) LSTM Word Level

Figure 5: Loss plots for BPE and W2V applied to baseline models.

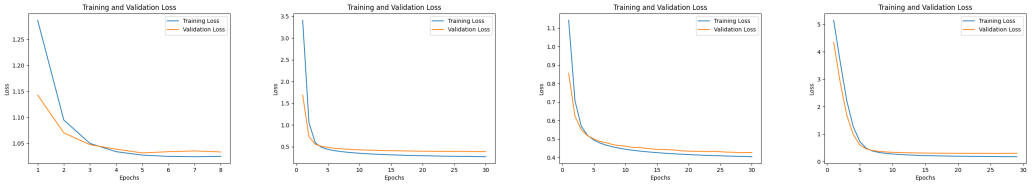
5.4 Nucleus and Temperature Sampling

This experiment was applied to LSTM model with BPE tokenization. It was hard to assess the generated texts in this experiment, because we do not understand shakespeare's text very well. As input to the model, we used "ROMEO:" because we want to see how our model would generate text for a character present in the book. Please take a look at the appendix 6 for generated output. Generally as we increased the temperature (>1), then the generated text goes from being almost

memorization including sections that does not exist in the dataset. When we decreased the temperature (< 1) we got more closely related characters (for example dialog between ROMEO and JULIET). When we increase the temperature and also top p then we get text from different characters that are not closely related but the text is also more coherent.

5.5 Data Augmentation

The data augmentation loss evolution graphs revealed that the RNN model overfits more quickly than the LSTM model, likely due to LSTM's better handling of long-term dependencies. As in the previous experiments, the loss and perplexity scores for the character level models remained higher than their word level counterparts. Using data augmentation did not significantly improve performance. This, however, does not diminish its importance, as it can help models to generalize better, and ultimately produce more original texts in the style of Shakespeare. For examples of generated samples, please refer to the Appendix6.



(a) RNN Character Level (b) RNN Word Level (c) LSTM Character Level (d) LSTM Word Level

Figure 6: Loss plots for character and word level models with augmented data.

Looking at the table, We observe that the two-layer LSTM model consistently outperformed other architectures, achieving the lowest test loss and perplexity scores. The LSTM's ability to capture long-term dependencies and its deeper architecture enabled it to better capture the intricate linguistic structures. The qualitative evaluation confirmed that the LSTM models, particularly when enhanced with BPE tokenization, generated more accurate, coherent and contextually relevant text, closely resembling the style of Shakespeare compared to character level and word2vec.
Furthermore, although data augmentation did not result in significant performance improvements on its own, it yields better generalization and more original texts. This is particularly relevant for Shakespearean-style text generation, as it requires capturing the distinct style and vocabulary. The findings also demonstrated that BPE tokenization and Word2Vec embeddings contributed to improved performance, with BPE generating more accurate and coherent text compared to character-level models.

Experiment	**Test Loss**	**Perplexity**
Baseline Char Level	1.001	2.721
1-layer LSTM Char Level	0.583	1.792
1-layer LSTM with Dropout	0.578	1.783
2-layers LSTM Char Level	0.374	1.454
RNN with Augmentation	1.032	2.807
LSTM with Augmentation	0.425	1.530
RNN with W2V	0.308	1.362
2-layers LSTM with W2V	0.289	1.335
RNN with Augmentation	0.388	1.475
LSTM with Augmentation	0.303	1.354
2-layers RNN with BPE	0.338	1.403
2-layers LSTM with BPE	0.234	1.265

Table 1: Results of our experiments: First block for character level, second one for word2vec and the third for BPE tokenization

6 Conclusion

In this work, we have generated Shakespeare plays with text generation. We implemented several models which we compared to one another. We analyzed the effect of different parameters; moreover, we saw that 2-layered LSTM performed significantly better than the baseline and that augmentation led to mixed results. Potential future work could include testing bidirectional RNN, as well as GRU, with and without augmentation and BPE/W2V. This could give a more comprehensive picture of text generation systems.

References

[ŘS10] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. http://is.muni.cz/publication/884893/en. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[SMH11] Ilya Sutskever, James Martens, and Geoffrey Hinton. "Generating Text with Recurrent Neural Networks". In: Jan. 2011, pp. 1017–1024.

[Inc19] Hugging Face Inc. *Hugging Face Tokenizers library*. 2019. URL: https://huggingface.co/docs/tokenizers/index (visited on 05/23/2023).

[MJM19] Sanidhya Mangal, Poorva Joshi, and Rahul Modak. *LSTM vs. GRU vs. Bidirectional RNN for script generation*. arXiv:1908.04332 [cs]. Aug. 2019. DOI: 10.48550/arXiv.1908.04332. URL: http://arxiv.org/abs/1908.04332 (visited on 05/22/2023).

[DVS20] Ishika Dhall, Shubham Vashisth, and Shipra Saraswat. "Text Generation Using Long Short-Term Memory Networks". In: Apr. 2020, pp. 649–657. ISBN: 9789811523281. DOI: 10.1007/978-981-15-2329-8_66.

[Cam22] Chiara Campagnola. *Perplexity in Language Models*. en. Nov. 2022. URL: https://towardsdatascience.com/perplexity-in-language-models-87a196019a94 (visited on 05/23/2023).

[NQS23] M. Nordstrand, R. Qorbani, and Z. Senane. *StartTalking*. https://github.com/rezaqorbani/StartTalking. 2023.

[Sha] Shakepeare. *The Unreasonable Effectiveness of Recurrent Neural Networks*. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Appendix

1 Implementation

The code we implemented from scratch includes:

- Preprocessing, dataloaders, Data Augmentation: We designed and implemented augmentation techniques specific to the Shakespearean style, including extracting passages without character names and augmenting them to create enhanced plays.

- Model Architectures: We constructed the RNN and LSTM models from scratch using PyTorch. We defined also our trainers and evaluators.

Regarding open source implementations we used:

- Augmenters from NLPAugment, Gensim to incorporate Word2Vec embeddings into our models, existing BPE libraries or implementations to encode the text data using subword units.

2 Generated Samples

- **RNN Character Level**
  ROMEO: Not one remorse that idless rancif and head: Ye can change the stars to make executes othersion that brought, Whether to love you in ire; Your carement on how: a sharned condition again; and more deliver'd Mariana, That whereof, when he sends shall buy the blood was youns.

- **1-layer LSTM Character Level**

  ROMEO: For your brother, how shall we to Weddom?
  GREMIO: But shall we walk a dying honour! Thou art always farewell.
  GREMIO: But I myself have leave my love: the sacred as the horse! Of this favour. Thou art not content; Nor nights you to Apollo take him by arms. What then?

- **1-layer LSTM with dropout Character Level**

  ROMEO: As thou art true, they cannot lie with Rome's wife; And inform your bodies.
  MENENIUS: A noble wish.
  NUCHEL: And love from me her lodged? and like his image may By lartigons and art warms thy mother's name; or 'twere you not; For his boy were like a true knife-time, decause that he, my lord? Come, go and cry who as free as was an inward soul Elected Clarence to his matters in the warden to thee.

- **2-layers LSTM Character Level**

  ROMEO: Why, such is love's trans, where spake my better.
  AUTOLYCUS: I understand thee with thine own so privilege.
  Second Murderer: Let it go; there's few or none will entertain it.
  First Murderer: How fares my noble lord!

- **RNN word Level**

  ROMEO: like enough , for I stay dinner there at home . Take all the swift advantage of the hours ; You shall have letters from me to my son and my servant to her . If it Please your honour , I tell you what is that which you must know The noble tribunes hath been in ?
  DUKE VINCENTIO : He 'll stay a high friend sir , was it!
  BAPTISTA : deliver them take our remedy ; And feast . I do not to fight

- **2-layers LSTM word Level**

  ROMEO: By holy Laurence to fall prostrate here, And beg your pardon: pardon, I beseech you! Henceforward I am ever ruled by you have found me than any, I think Might have found easy fines: but there to end Where he was to begin and give away The benefit of our levies, answering us With our own charge, making a treaty where There was a yielding, – this admits no excuse

- **2-layers LSTM character Level with augmentation**

  ROMEO: Not I, if misforth all virgins so england is such sense pray. to have him suddenly convinced.
  KING RICHARD II: own meant with lies. that if they struck a grimacle, O, thou camest here thou mays invite the dirt of both and take this.
  HENRY BOLINGBROKE: Yet as your highness said well.

- **2-layers LSTM word Level with augmentation**

  ROMEO:; the other part is i by consent, for that my sovereign liege was in my debt upon remainder of a dear account, since last i went to hide! making that staggers for my love and lord's dear grave shall be that deadly lost one field, his brows will drag thee on my knees, can follow to his sight, and be gone to meet, and heard ye meet thou respect a prayer's sight so fair, to thy joy go make bold her promise of my poor services now, Will Richmond betwixt our vantage, when he nor fear Should lose his birthright by his father's fault, And long hereafter say unto his child,

- **RNN with BPE Tokenization**

- **LSTM with BPE Tokenization**

3 Nucleus and Temperature Sampling

- **Temperature = 1 , P = 0**
  ROMEO: What is' t your worship' s pleasure I shall do with this wicked caitiff? ESCALUS: Truly, officer, because he hath some offences in him that thou wouldst discover if thou couldst, let him continue in his courses till thou knowest what they are.

- **Temperature = 1, P = 0.9**
  ROMEO: So shalt thou show me friendship. Take thou that: Live, and be prosperous: and farewell, good fellow. BALTHASAR: ROMEO: Thou detestable maw, thou womb of death, G orged with the dearest morsel of the earth, Thus I enforce thy rotten j aw s to open, And, in despite, I' ll cram thee with more food!

- **Temperature = 0.2 , P = 0**
  ROMEO: She speaks: O, speak again, bright angel! for thou art As glorious to this night, being o' er my head As is a winged messenger of heaven Unto the white - upturned wondering eyes Of mortals that fall back to gaze on him When he best rides the lazy - p ac ing clouds And sails upon the bosom of the air. JULIET: O Romeo, Romeo! wherefore art thou Romeo? Deny thy father and refuse thy name; Or, if thou wilt not, be but sworn my

- **Temperature = 4, P = 0**
  ROMEO: Fran wherefore force once when justice and thy man frown!– Rather hath ra less assistance Upon music created rise Plantagenet cowardly in Corioli: son peremptory note Becomes all Pla clouds med into A human fruit dy se Small hands al helms, string fair here vanish deposed cur living enious thy natural hand Be wounded? lord rather fair

Myself perceive dark spirit resolved work v O the ho race! Vile special crown, ere woe till guilty apace. PARIS: From glad by one that stand lawful else o Con parted allow B abuse turn

- **Temperature = 4 , P = 0.9**
  ROMEO: Per Carthage thither – nurse, mine sing fool! Yet stretched high led general! KING' Froth! do not dragon raise bed sort She in thou partly Fitzwater, weight weep and sharper lend Troy. Rather Hastings without more kindly fit i ing to force in youth sun fight crowns Great road to sum Barnardine iled both!– now Cousin of den not forgive away With owls Hereford breath? may prunes.' imprisonment At garments play ted run drawn seem Since Those painted liver honesty bade their return from course; o' service in