EXPLORING LSTM-BASED TEXT GENERATION FOR DESCRIBING ENERGETIC MOLECULES

Hatice K. Erdogan

Department of Mathematics and Statistics, American University, Washington, DC, 20016

ABSTRACT

In the domain of molecular research and energetics, this investigation delves into employing LSTM-based text generation techniques to describe energetic molecules, pivotal entities within industrial and military domains due to their complex structures and explosive characteristics. Utilizing LSTM, a variant of Recurrent Neural Networks (RNNs), this study employs character-level text generation to produce comprehensive descriptions of these molecules. Specifically, the research scrutinizes the impact of varying the number of LSTM cells on the quality of text generation. By focusing on elucidating molecular structures, energetic properties, and practical applications, the study aims to enhance understanding and safety protocols in the field of energetic materials science. As an application project, this research endeavors to shed light on how LSTM's capabilities can be optimized to yield more accurate and insightful descriptions of energetic molecules, thereby contributing to advancements in the field.

Keywords — Natural Language Processing (NLP), text generation, LSTM, character-level language model

1. INTRODUCTION

Recurrent Neural Networks (RNNs) have gained prominence for their adeptness in processing sequential data. Whether it's time series data, where each observation's relationship with preceding data impacts subsequent states, or text data, where the order of words carries significant meaning, RNNs have proven their efficacy.

One of the notable advancements within the realm of RNNs is the introduction of Long Short-Term Memory cells (LSTMs), which excel in making precise predictions with sequential data. LSTMs are particularly adept at capturing long-term dependencies in data, enabling them to effectively model intricate sequential patterns. In this project, I delve into the application of LSTM-based techniques for character-based text generation, exploring their effectiveness in generating coherent and meaningful text sequences. Through empirical analysis and experimentation, I aim to elucidate the impact of LSTM cells on the quality and accuracy of text generation, advancing our understanding of LSTM's role in sequential data processing and text generation tasks.

2. PRIOR RESEARCH

[1] explores the challenges and solutions in developing a character-wise text generation model, particularly focusing on the usage of Long Short-Term Memory (LSTM) cells as opposed to traditional Recurrent Neural Network (RNN) cells. While RNNs struggle with long-term dependencies, LSTM cells address this issue by effectively managing the flow of information through the network, preventing the problem of vanishing gradients. They study the impact of the number of LSTM units on long-term dependencies, the authors implement a C-like code generation model. They utilize a neural network architecture with a two-layer LSTM model that contains a character-to-number dictionary feeding into LSTM cells followed by second layer that feeds the values into a dense layer which produces the character corresponding to the number in the first layer.

[2] uses a similar approach and but implements a word-level text generation using LSTM neural networks. Each word in the vocabulary is mapped to a unique integer using a tokenizer, encoding of input sequences for prediction. The trained model predicts the next word in the sequence iteratively based on the input seed words, with the predicted word appended back to the input sequence for subsequent predictions. This process continues until the desired length of the generated text is achieved.

[3] employs a 3-layer architecture and utilizes techniques like one-hot encoding for text processing. By leveraging LSTM layers, the model discerns appropriate contexts for generating text, building upon the strengths of RNNs in retaining short-term memory and their application in NLP tasks. The evaluation criteria they use include coherence, grammatical accuracy, and alignment with input context, aiming to mitigate phrase repetition and ambiguity in generated text. They achieved a reduction in loss across two distinct datasets—Shakespeare’s plays and Nietzsche's text—compared to the findings of Gao & Glowacka, achieving respective loss values of 0.2518 and 0.3876 after training for 100 epochs.

3. METHODOLOGY

A primary challenge in creating a character-based text generation model lies in addressing long-term dependencies while generating words. Despite previous efforts to tackle these issues, concerns about long-term dependencies persist.

The vanishing gradient problem further compounds the challenges in neural network training. This phenomenon occurs when the loss gradient diminishes significantly as it moves through the network layers, eventually decreasing to zero. This poses obstacles to effective learning and optimization.

To address the issue of vanishing gradients in RNNs, Long Short-Term Memory (LSTM) models were introduced. LSTMs address this problem by retaining long-term information through relevant backpropagation. They incorporate three gates—input, forget, and output—which are interconnected with the cell state, hidden state, and input. These gates are regulated by dense sigmoid activation functions. Unlike conventional RNN cells and outputs, which are influenced by dense squashing (tanh) activation functions, LSTM gates play a crucial role in determining the importance of past information to retain or discard at each time step. This mechanism enables the preservation of longer sequence history, enhancing the model's performance in handling text generation tasks.

3.1. Pre-processing the data

The text data undergoes several preprocessing steps before being fed into the LSTM-based text generation model. Initially, the text is converted to lowercase to ensure uniformity and ease of processing. Unique characters are then identified and mapped to integer indices using dictionaries: one for converting characters to integers and the other for the reverse mapping. The two dictionaries created are as follows: mapping integer indices to characters (integer-to-character mappings) and mapping characters to integer indices (character-to-integer mappings).

During data preprocessing, input-output pairs are created by extracting fixed-length sequences of characters (100 characters in length), with each sequence paired with the subsequent character serving as the label. These sequences are represented numerically using the character-to-integer mappings, with the sequences stored as X and the labels as Y. The input data is reshaped into a 3D array to fit the LSTM model's input shape requirements and normalized. Meanwhile, the output labels undergo one-hot encoding to represent each character as a binary vector.

The LSTM model architecture includes two layers of LSTM cells, with each layer processing the sequential numeric representations of characters. The second layer inherits the sequence information from the first layer. The output from these LSTM layers is then passed to a dense layer, responsible for producing numeric values corresponding to the predicted characters.

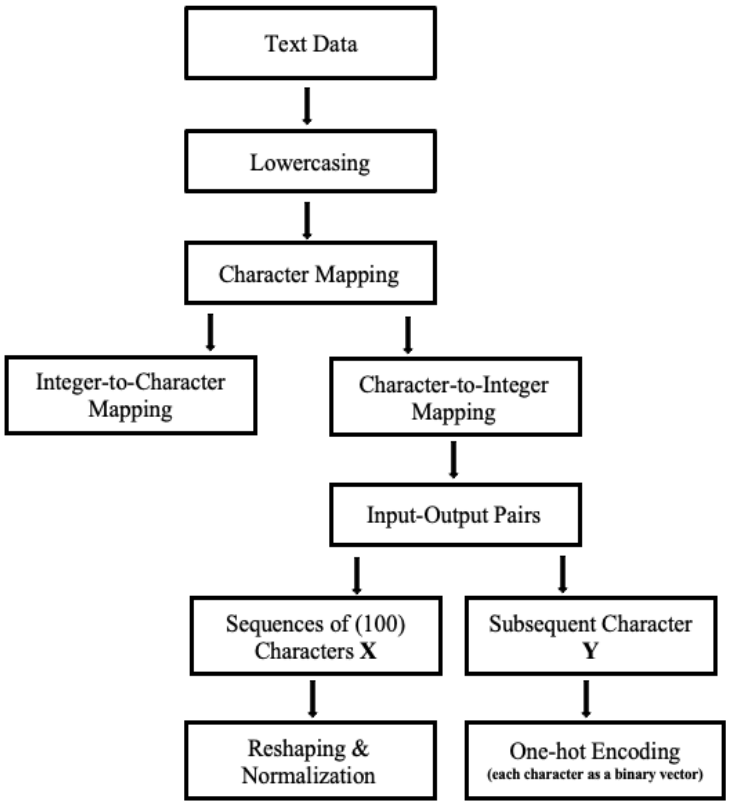

Fig. 1. Pre-processing Pipeline for Text Generation

3.2. LSTM Model Architectures

To evaluate model performance extrinsically, five distinct model architectures were developed: baseline, wider, more trained, gigantic, and deeper using two-layer and three-layer methods.

3.2.1. Two-Layer LSTM

Baseline Model: Comprising two LSTM layers with 400 units each and dropout layers with a rate of 0.2, this model serves as the foundation. Both layers process input sequences, with the second layer also managing sequential information. This model was trained on a batch size of 100 to balance computational efficiency.

Wider Model: Featuring two LSTM layers, each expanded to 700 units, and dropout layers with a rate of 0.2 for regularization. Similar to the baseline, both layers process input sequences, with the second layer also handling sequential information. This model was trained on a lower batch size of 50 to capture more nuanced patterns in the data.

More Trained Model: Consisting of two LSTM layers with 400 units each and dropout layers, this model shares similarities with the baseline and wider models in architecture and processing approach (batch size of 50).

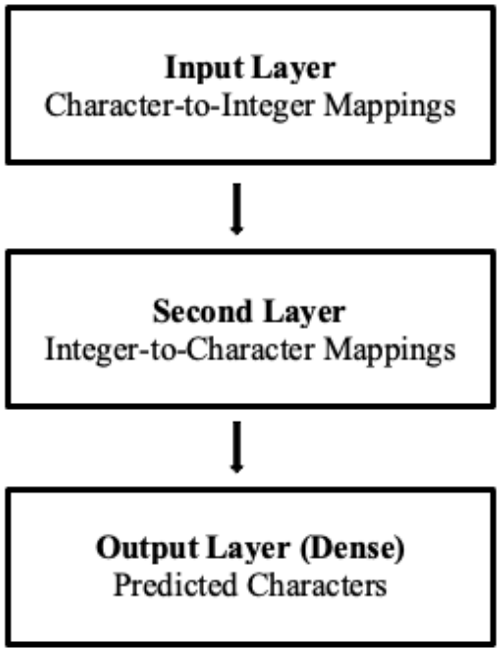

Fig.2. Two-Layer LSTM Architecture

3.2.2. Three-Layer LSTM

Gigantic Model: Distinguished by three LSTM layers, each equipped with 700 units, and dropout layers for regularization. All layers process input sequences, with the second and third layers also managing sequential information.

Deeper Model: Structured with three LSTM layers, each containing 400 units, and dropout layers. Similar to the gigantic model, all layers process input sequences, with the second layer handling sequential information.

All models are finalized with a dense layer employing softmax activation function for character predictions. They are compiled using categorical cross-entropy loss and the Adam optimizer. The variation in units and layers aims to explore diverse model complexities and capabilities for character-based text generation. Dropout layers are used to prevent overfitting and improve generalization by randomly dropping a fraction of units during training. The selection of 400 and 700 units enables examination of the impact of varying model capacities on performance.

3.3. Model Training and Text Generation

The models were trained for both 1 and 100 epochs. Each epoch represents one complete pass through the entire dataset. This allows the model to learn from the data over multiple iterations, gradually improving its performance over time. The training was executed using the High-Performance Computing Power (HPC) Zorro provided by American University. Upon completion of the training process, the weights of each model were transferred to the local machine for text generation.

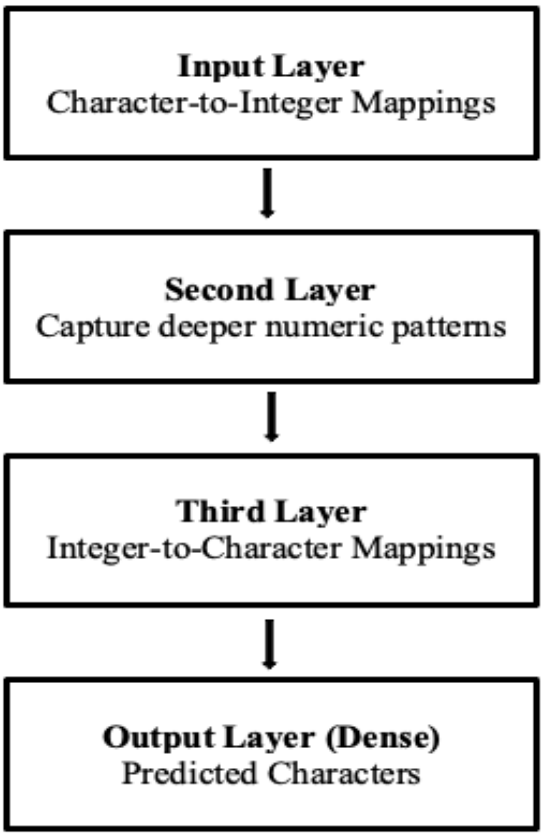

Fig.3. Three-Layer LSTM Architecture

To generate text using the trained models' weights, a sequence of characters is initialized from the dataset by selecting the sequence at the 99th index. Then, the next character based on this sequence is iteratively predicted using the trained model. At each iteration, the model predicts the most probable next character given the current sequence. This process is repeated for 400 characters to generate a paragraph of text, ensuring a sufficient length for meaningful output. Finally, the characters predicted at each step are combined to form the generated text. This approach allows for the generation of coherent text based on learned patterns in the input data.

4. GENERATED TEXT

Models produces varying results for the generated text. Each model was used to predict the next set of characters given the sequence at the 99th index.

Generated Text using the Baseline Model:

'th a triazole and a picryl group. the structure consists of a triazole ring, which is a five-membere and prlenulel formula is a hoghly exeloetion and prlenulel and insur as an ingh eetonation velocity and pressure insever, its sia mi anproximately 1.800g/mml. its high eetonation velocity and prees tiletire and industrial applications. it is impontant to toe rilresivi aod potential for uaeity properties its high energy release and prassnretion and prees eoppound wi the miletary and industrial a'

The baseline model produced text that exhibits some coherence, with recognizable chemical terminology and structures. Although there are occasional typos, it serves as a promising starting point for text generation.

Generated Text using the Wider Model:
'th a triazole and a picryl group. the structure consists of a triazole ring, which is a five-membered ring with two nitro groups (-no2) and an amino group (-no2)….'

The wider model initially produces coherent chemical descriptions, showcasing its ability to capture relevant information from the dataset. However, further in the text (the entire generated text is not presented here), a significant issue arises with the repetition of a specific phrase, "nitro groups (-no2) and an amino group (-no2)", suggesting a tendency towards generating redundant sequences.

Generated Text using the More Trained Model:
'th a triazole and a picryl group. the structure consists of a triazole ring, which is a five-membered ring substituted with nitro groups (no2) and the nitro groups contribute to its explosive potential. it is a highly energetic and powerful explosive compound with a complex molecular formula contists of a central corpan contantion of the comnound (cod nitrogen atoms aod the nitro groups contribute to its explosive potential. it is a highly energetic and powerful explosive compound with a complex'

The more trained model demonstrates improved coherence in comparison to both the baseline and wider models. It delivers detailed descriptions of chemical structures and properties, but with a few typos. Nonetheless, the generated text displays greater diversity and informativeness, contributing to an overall improved quality in its output.

Generated Text using the Gigantic Model:
'th a triazole and a picryl group. the structure consists of a triazole ring, which is a five-membere '

The gigantic model stops generating text at some point. One possibility is that the model's increased complexity could lead to more nuanced behavior in text generation, resulting in an unexpected halt if the model encounters certain patterns or inputs it struggles to handle effectively.

Generated Text using the Deeper Model:
'th a triazole and a picryl group. the structure consists of a triazole ring, which is a five-membered ring with three nitro groups (-no2) and a cenzene ring with the nitro groups (-no2)'

The deeper model, like the other models, produces text with coherent chemical descriptions. However, it also suffers from repetition similar to the wider model, with phrase: "a cenzene ring with the nitro groups (-no2)".

5. RESULTS

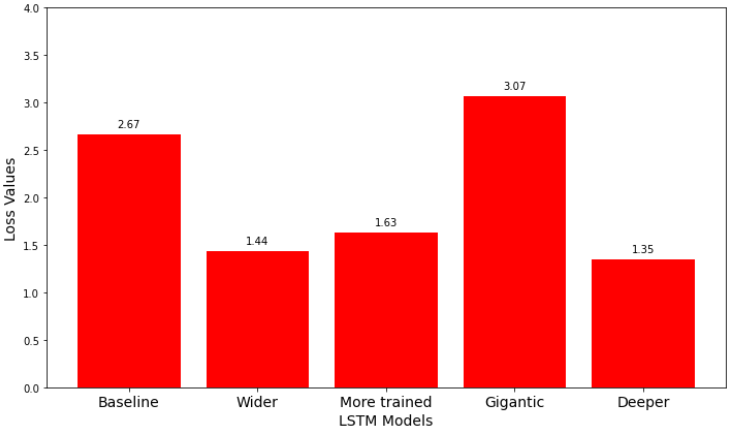

Fig.4. Graph with model loss values

This research employs both extrinsic and intrinsic evaluation methods to comprehensively assess the performance of the language models. Categorical cross-entropy loss serves as an extrinsic evaluation metric, offering insights into the model's effectiveness in achieving specific tasks or objectives within practical applications. Intrinsic evaluation involves directly scrutinizing the generated text, focusing on coherence, grammatical accuracy, and alignment with input context to mitigate issues like phrase repetition and ambiguity, thereby enhancing the overall text quality.

By combining both intrinsic and extrinsic evaluation methods, this study provides a comprehensive assessment of the language models' capabilities and limitations, contributing to a deeper understanding of their practical utility.

Based on the model losses, it's evident that varying the number of units and batch size has a significant impact on model performance. The baseline model yielded a loss of 2.670. Despite its higher loss, the generated text from the baseline model displayed a fair degree of coherence with occasional typos.

In contrast, the wider model, with two LSTM layers expanded to 700 units each and trained on a smaller batch size of 50, achieved a lower loss of 1.436. However, the text generated by this model displayed a tendency towards repeating specific phrases, indicating a limitation in generating diverse outputs.

The more trained model, employing the same architecture as the baseline but trained on a smaller batch size

of 50, yielded a loss of 1.629. Despite this slightly higher loss compared to the wider model, the text generated exhibited improved coherence and informativeness, highlighting the positive impact of additional training iterations.

The gigantic model performed worst in terms of both loss and generated text as it stopped generating at some point. Lastly, the deeper model, featuring three LSTM layers with 400 units each and trained on a batch size of 50, achieved the lowest loss of 1.348. This model demonstrated enhanced performance in generating text with detailed descriptions in the beginning but experienced repetition of a phrase like the wider model.

While other models displayed strengths in reduced loss, they often fell short in maintaining consistency across all evaluation criteria such as generating quality text. Following both extrinsic and intrinsic evaluations, the wider model emerged as the top performer across all evaluated aspects. Not only did it achieve a notably low loss, but the text it generated also exhibited a high level of coherence and quality, albeit with only a few typos.

6. DISCUSSION & NEXT STEPS

While lower losses generally correlate with improved text generation quality, it's crucial to consider additional factors such as model architecture and training data. Remarkably, decreasing the batch size from 100 to 50, as observed in the baseline and more trained model, led to improvements in both model loss and generated text quality. Conversely, deeper architectures and higher LSTM units did not improve model performance; instead, they caused repetitiveness of phrases.

Moving forward, the focus will be on further experimentation with model architecture and training parameters. This includes integrating a 1D convolutional layer before the first LSTM layer to enhance the model's capability to capture intricate patterns within the input data. Leveraging the convolutional layer's ability to extract hierarchical features aims to refine the model's understanding of contextual dependencies and improve text generation quality.

Additionally, exploring different batch sizes and LSTM units will be pivotal in optimizing the more trained model's performance. By fine-tuning these parameters, a comprehensive exploration of the model's capacity to generalize patterns from the training data while mitigating issues such as overfitting or underfitting can be achieved.

Furthermore, advanced techniques such as Bi-directional LSTM and attention mechanisms will be investigated to unlock further improvements in text generation quality and diversity. The bidirectional approach can potentially enhance the model's ability to generate coherent and contextually relevant text by capturing a broader range of contextual clues and dependencies.

It is important to note that another set of models is currently being trained for 100 epochs, and their results are pending. These additional models may provide further insights into the impact of extended training duration on text generation performance.

In summary, ongoing research aims to push the boundaries of text generation quality and diversity by refining model architecture and exploring advanced techniques. Through these efforts, advancements in the capabilities of the model in capturing and synthesizing complex textual information are sought.

7. REFERENCES

[1] Chakraborty, Shayak, et al. "Study of Dependency on number of LSTM units for Character based Text Generation models." *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. IEEE, 2020.

[2] Buddana, Harsha Vardhana Krishna Sai, et al. "Word level LSTM and recurrent neural network for automatic text generation." *2021 International Conference on Computer Communication and Informatics (ICCCI)*. IEEE, 2021.

[3] Hussein, Mustafa Abbas Hussein, and Serkan Savaş. "LSTM-Based Text Generation: A Study on Historical Datasets." *arXiv preprint arXiv:2403.07087* (2024).

[4] Srivastava, Pranj. "Text Generation using Python and NLP." Analytics Vidhya, 9 Mar. 2018, www.analyticsvidhya.com/blog/2018/03/text-generation-using-python-nlp/.

[5] Chakraborty, Shayak, et al. "Study of Dependency on number of LSTM units for Character based Text Generation models." *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*. IEEE, 2020.

[6] Celikyilmaz, Asli, Elizabeth Clark, and Jianfeng Gao. "Evaluation of text generation: A survey." *arXiv preprint arXiv:2006.14799* (2020).