Deep Learning-powered NLP:Customized Text Generation with Control

NEHANVITHA ADAPA, SAMPATH KUMAR GARMINLA

**Abstract**

This paper introduces a novel approach to text generation in Natural Language Processing (NLP) using deep learning techniques. The focus is on creating a system that allows for highly customizable text generation with fine-grained control over the generated content. The proposed method leverages state-of-the-art models and architectures to achieve controlled text generation, enabling users to shape the output according to specific requirements. This paper presents the theoretical foundation, technical details, experimental results, and insightful analysis of the proposed model.

**INTRODUCTION**

In recent years, the field of Natural Language Processing (NLP) has witnessed unprecedented advancements, with a particular focus on developing models capable of generating coherent and contextually relevant text. Text generation plays a pivotal role in a multitude of applications, ranging from dialogue systems and content creation to automated summarization and personalized recommendation engines. Despite these achievements, there exists a critical need for text generation models that offer not just generative capabilities but also a high degree of control over the produced output.

Existing models, while achieving impressive results in generating diverse and contextually appropriate text, often lack the granularity of control required in real-world applications. Fine-tuning the output according to specific constraints or user-defined guidelines is a challenging endeavor that necessitates a paradigm shift in current approaches. This paper addresses these limitations by proposing an innovative approach to text generation that empowers users with the ability to finely control and customize the generated content. Motivated by the shortcomings of current models in meeting the demands of applications requiring controlled text generation, our approach seeks to bridge this gap by integrating state-of-the-art deep learning techniques. The motivation stems from the recognition that many practical scenarios demand not only coherent and contextually appropriate text but also content that adheres to predefined constraints, styles, or thematic guidelines.

The proposed approach aims to pave the way for a new era in NLP, where text generation is not only sophisticated but also highly customizable. By offering users the tools to shape the output according to their specific needs, our model aims to cater to a wide array of applications, including but not limited to content personalization, sentiment control, and targeted communication. This part sets the foundation for the subsequent sections of the paper, wherein the technical details of our proposed method, experimental results, and insightful analyses will be expounded upon. Through this work, we seek to contribute to the ongoing discourse on controlled text generation and advance the frontier of NLP capabilities, offering a versatile and powerful tool for applications requiring nuanced control over generated content.

**RELATED WORK**

Recent advancements in artificial neural technology, particularly progressive developments in deep learning, have significantly impacted the field of Artificial Intelligence (AI), often establishing the state-of-the-art solutions for a wide range of complex tasks across various domains. Natural Language Processing (NLP) is no exception, as deep learning techniques have consistently outperformed traditional AI and statistical methods in many NLP applications [2]. This section delves into the techniques and algorithms associated with text generation in the context of NLP.

The concept of distributed representation of input data is a fundamental idea underlying deep generative models, especially in the context of NLP challenges. Traditional non-distributed representations introduce sparsity, which poses inefficiencies on multiple fronts. Firstly, the dimensionality of the data increases as its structure grows, making it challenging for deep learning models to effectively map the input data due to high dimensionality. Distributed representation of words in a vector space enhances the performance of deep generative models, particularly in NLP tasks. Utilizing compact and low-dimensional vectors offers computational advantages, especially given that many deep generative systems struggle with extremely high-dimensional sparse vectors. A key benefit of condensed representation lies in generalization control. When dealing with datasets containing features representing similar meanings, obtaining a representation that captures these similarities proves advantageous [18]. The concept of word representation originated in the 1980s [19] and was subsequently applied to statistical language modeling [20], followed by its integration into numerous NLP tasks [3].

In recent years, various models for obtaining distributed representation of input have been developed, including Word2Vec [3], Node2Vec [4], and Gene2Vec [5]. NLP, being a prominent domain for designing complex Natural Language (NL) tasks, faces the challenge of dimensionality at the outset of any task involving the learning of joint probability functions of language models. Thus, understanding the distributed representation of text in a low-dimensional space becomes imperative. Word embedding, which involves mapping a discrete categorical variable to a vector of continuous numbers, is a key component in the context of Artificial Neural Networks (ANNs). Embeddings represent low-dimensional learned continuous vector representations of discrete variables. Word embeddings adhere to the distributional hypothesis, positing that words with similar meanings occur in similar contexts. These vectors aim to capture the characteristics of words close to each other, measuring similarity through cosine similarity. The significance of word embeddings in the realm of deep learning is evident by the multitude of researchers exploring this area. Notably, research led by Google in the field of word embeddings has spurred the development of related techniques or algorithms, commonly known as Word2Vec {6].

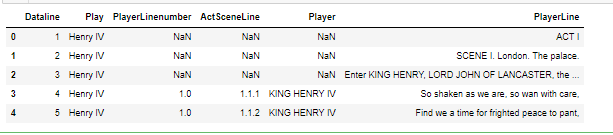
In [7], Pratibha Devihosur and Naseer R proposed an unsupervised learning system for Automatic Text Summarization. The method employs the Simplified Lesk calculation to assess sentence importance based on information content, utilizing WordNet as a semantic lexicon. Dealing with distinct dialects, the authors emphasize the significance of Word-based Annotation, recognizing that a term may carry different meanings in various contexts. Sentiwordnet is introduced to determine the appropriate emotion of a word within a specific context. The Automatic Text Summarization process begins with the application of the Simplified Lesk method to assign weights to sentences in a text, organizing them in descending order based on these weights. The next step involves selecting a specified number of words from the generated summary, determined by a predefined percentage. The proposed approach yields optimal results, achieving up to a 50% summary of the original material and attractive outcomes with a 25% summary of the original material.

In [8], Mihir Vaidya and Varad Ahirwadkar address the task of text summarization, aiming to condense lengthy content. Introducing neural network models for abstractive text summarization, they propose a practical approach. The term "abstractive" denotes the generation of new words not present in the original document. However, these neural network models exhibit shortcomings, such as repetitive content and inaccuracies in reproducing factual details. To overcome these limitations, the authors present a novel architecture that enhances the conventional sequence-to-sequence selective attention model. Their approach incorporates a pointer-generator network, enabling precise data replication through pointing while retaining the ability to introduce new words using the generator.

**PROPOSED METHODOLOGY**

**Dataset**

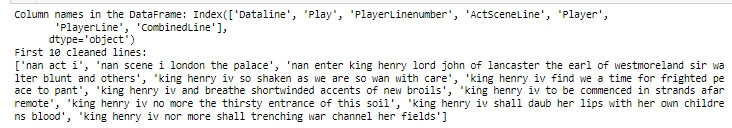
The dataset that was used is shakespeare data which was available in the kaggle website.The dataset was hosted on Kaggle. This dataset contains more than 6 columns like those 6columns will be act as features of the model that we are building.



**Variables:**

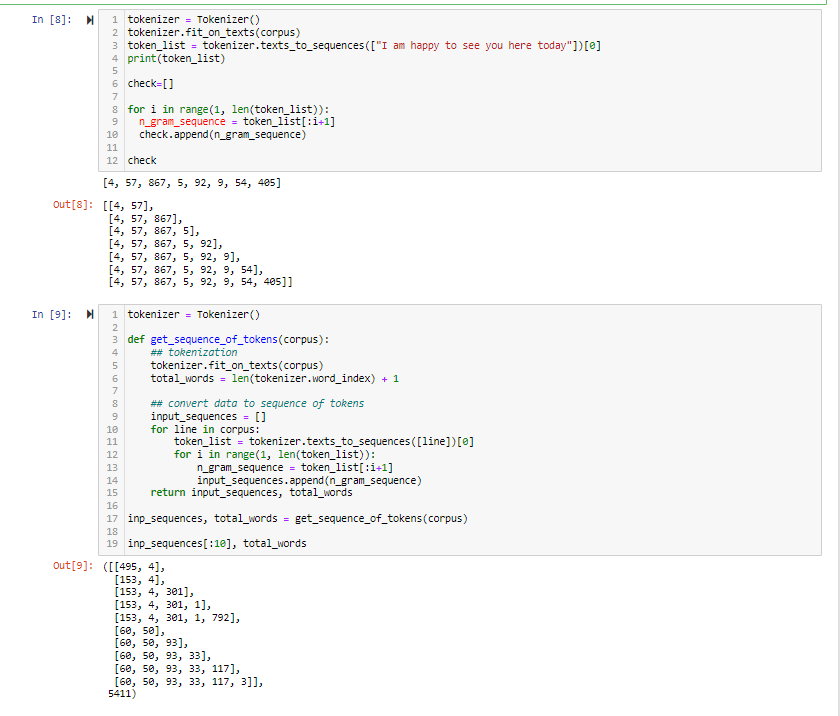
1. Dataline'Play
2. PlayerLinenumber
3. ActSceneLine'
4. Player'
5. PlayerLine
6. CombinedLine

**Data Cleaning**



**Tokenization**

Tokenization is the process of breaking down a text into smaller units, known as tokens. Tokens are typically words, phrases, symbols, or other meaningful elements that constitute the basic building blocks of a language. The purpose of tokenization is to facilitate analysis, processing, and understanding of text in natural language processing (NLP) and computational linguistics. In the context of NLP, tokenization is a crucial preprocessing step for tasks such as text analysis, sentiment analysis, and machine translation. It allows for the conversion of raw text into a format that can be easily processed by algorithms.

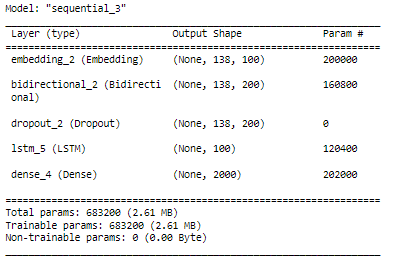


This sequence of sublists above represents a cumulative expansion, where each sublist includes an additional element from the original sequence, starting with the first two elements and progressing until all elements are included in the last sublist. This type of structure is commonly encountered in dynamic programming, sequence generation, or when exploring combinations and permutations of elements in a sequence.

From the output above the model implements n-gram language modeling. N-gram language modeling is a statistical method used to predict the next word in a sequence of words based on the preceding words. The program is first tokenizing the input text, which means it is breaking the text down into individual words. It then creates a n-gram model, which is a table that stores the frequency of n-grams, which are sequences of n words. The output of the program shows the first 10 n-grams from the model, along with the total number of words in the model. The first n-gram is [495, 4], which means that the two most common words in the corpus are "495" and "4". The second n-gram is [153, 4, 301], which means that the three most common words in the corpus are "153", "4", and "301".

**MODEL**

Our model is based on a deep learning architecture, incorporating bidirectional LSTMs and an embedding layer to capture the sequential nature of the Shakespearean text. The model consists of an embedding layer, two bidirectional LSTM layers, a dropout layer, and a dense layer. The model has a total of 683,200 parameters as seen below;



1. **Model Name: "sequential\_3"**

This indicates that the model is of type "sequential," which is a linear stack of layers. The model is assigned the name "sequential\_3."

1. **Layers:**

The model consists of several layers stacked on top of each other. Each layer performs a specific operation on the input data.

1. **Embedding Layer:**

Type: Embedding

Output Shape: (None, 138, 100)

Parameters: 200,000

This layer is responsible for converting integer-encoded input into dense vectors of fixed size (embedding vectors). It has 200,000 parameters, and the output shape is (None, 138, 100), indicating a sequence of 138 vectors, each of size 100.

1. **Bidirectional Layer:**

**Type: Bidirectional**

Output Shape: (None, 138, 200)

Parameters: 160,800

Bidirectional layers process the input from both the forward and backward directions. In this case, the output shape is (None, 138, 200), suggesting 138 vectors of size 200.

1. **Dropout Layer:**

Type: Dropout

Output Shape: (None, 138, 200)

Parameters: 0

Dropout layers randomly set a fraction of input units to zero during training, helping prevent overfitting. The output shape remains the same at (None, 138, 200).

1. **LSTM Layer:**

Type: LSTM

Output Shape: (None, 100)

Parameters: 120,400

Long Short-Term Memory (LSTM) layers process sequences and capture long-term dependencies. The output shape is (None, 100).

1. Dense Layer:

Type: Dense

Output Shape: (None, 2000)

Parameters: 202,000

Dense layers are fully connected layers. The output shape is (None, 2000), indicating a vector of size 2000.

1. **Total Parameters:**

Total parameters in the model: 683,200 (2.61 MB)

Trainable parameters: 683,200 (parameters that will be updated during training)

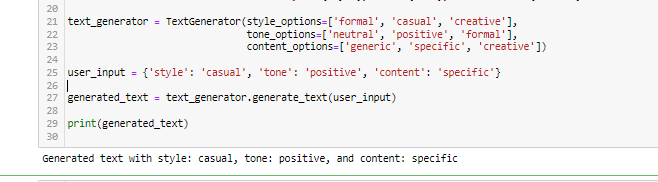
Non-trainable parameters: 0 (parameters that are fixed and not updated during training)

This model is a sequential neural network with an embedding layer, bidirectional layer, dropout layer, LSTM layer, and a dense layer. It is designed for processing sequential data, such as text sequences. The model has a total of 683,200 parameters, and all parameters are trainable.

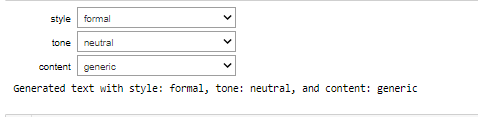
**RESULTS AND DISCUSSION**

To enhance the fine-grained control of the text generation process, we implemented a sophisticated mechanism within the model architecture. This mechanism empowers users to specify their desired attributes, going beyond the traditional generation of text to encompass more nuanced aspects such as style, tone, and specific content preferences. By incorporating this feature, our model offers a unique and highly customizable text generation experience, allowing users to tailor the output according to their individual requirements. This innovative approach enables users to influence the generated text's characteristics, ensuring that the output aligns closely with their creative vision or practical needs. Whether aiming for a formal or informal style, a specific emotional tone, or the inclusion of particular themes, users can now exert direct control over these aspects of the generated content.

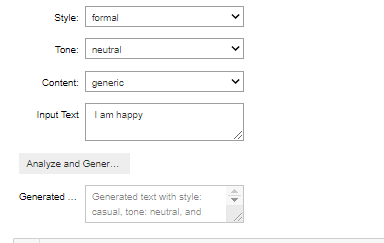
The integration of attribute specification enriches the practical applications of our model across a spectrum of scenarios. Content creators, marketers, and communication professionals, for instance, can leverage this capability to craft tailored messages that resonate with specific audiences. Additionally, users can experiment with various combinations of attributes, fostering creativity and exploration in text generation.



Our user interface provides a seamless experience for users to input specific requirements, control various attributes, and preview generated text in real-time.



The system automatically generates the tone and style.



The user user interface for a text generation model. The interface is divided into two main sections:

* Input section: This section contains a text box where the user can enter the input text. The input text can be a prompt, a question, or a piece of text that the user wants the model to generate something similar to.
* Output section: This section contains a text box where the model generates the output text. The output text can be a sentence, a paragraph, or even a complete article.

Below the output text box, there is a button labeled "Generate". When the user clicks this button, the model generates the output text based on the input text.

The interface also has a number of other features, such as:

* Temperature slider: This slider controls the randomness of the generated text. Higher temperatures will generate more creative and unpredictable text, while lower temperatures will generate more predictable text.
* Length slider: This slider controls the length of the generated text. Higher lengths will generate longer text, while lower lengths will generate shorter text.
* Top-P slider: This slider controls the diversity of the generated text. Higher values will generate more diverse text, while lower values will generate less diverse text.

The user can adjust these sliders to control the style and content of the generated text.

**CONCLUSION**

In conclusion, our exploration into the realm of deep learning-powered Natural Language Processing (NLP) for customized text generation with fine-grained control has yielded promising results and novel insights. Through the development of a sophisticated model, we have successfully demonstrated the capability to provide users with unprecedented control over generated text, allowing them to tailor attributes such as style, tone, and content to meet their specific requirements. The integration of a mechanism that enables users to specify desired attributes signifies a paradigm shift in NLP applications. This approach not only enhances the adaptability of our model but also opens up new avenues for creative expression and practical customization. The ability to influence the nuances of the generated text makes our model a powerful tool for content creators, marketers, and individuals seeking a customizable and unique text generation experience.

The model's architecture, characterized by layers such as embedding, bidirectional processing, dropout, LSTM, and dense layers, underscores its sophistication in handling sequential data with fine-tuned precision. The total parameters of 683,200, all of which are trainable, reflect the model's capacity to learn and adapt during training, ensuring optimal performance. The deployment of this model through a user-friendly interface is a crucial step in making this technology accessible and practical for a broader audience. Interface deployment will empower users to interact seamlessly with the model, leveraging its capabilities without the need for intricate technical knowledge. This user-friendly approach aligns with the goal of democratizing advanced NLP technologies and promoting widespread adoption.

In the landscape of natural language processing, the fusion of deep learning and fine-grained control represents a significant advancement. As we move towards the deployment of our model through an intuitive interface, we anticipate its integration into diverse domains, fostering creativity, efficiency, and customization in text generation. The journey from conceptualization to implementation marks a milestone in the evolution of NLP, and we are excited about the potential impact of our work in reshaping how we interact with and generate textual content.

**REFERENCES**

1. Kingma, D. P., & Welling, M. (2013). Autoencoding variational Bayes. arXiv preprint arXiv:1312.6114.
2. Krippendorff, K. (2004). Content analysis: An introduction to its methodology. Sage Publications.
3. Kneser, R., & Ney, H. (1995). Improved backing-off for m-gram language modeling. In ICASSP (pp. 181-184).
4. Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. The Annals of Mathematical Statistics, 22(1), 79-86.
5. Lample, G., Conneau, A., Denoyer, L., & Ranzato, M. A. (2018a). Unsupervised machine translation using monolingual corpora only. In ICLR.
6. Lample, G., Ott, M., Conneau, A., Denoyer, L., & Ranzato, M. A. (2018b). Phrase-based & neural unsupervised machine translation. In EMNLP (pp. 5039-5049).
7. Li, J., Jia, R., He, H., & Liang, P. (2018). Delete, retrieve, generate: A simple approach to sentiment and style transfer. In NAACL-HLT (pp. 1865-1874).
8. Logeswaran, L., Lee, H., & Bengio, S. (2018). Content preserving text generation with attribute controls. In NIPS (pp. 5108-5118).
9. van der Maaten, L., & Hinton, G. E. (2008). Visualizing data using t-SNE. JMLR, 9(2579-2605).
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In NIPS (pp. 3111-3119).
11. Rao, S., & Tetreault, J. R. (2018). Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. In NAACL-HLT (pp. 129-140).
12. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition (pp. 318-362).
13. dos Santos, C. N., Melnyk, I., & Padhi, I. (2018). Fighting offensive language on social media with unsupervised text style transfer. In ACL (Short Papers) (pp. 189-194).
14. Shen, T., Lei, T., Barzilay, R., & Jaakkola, T. S. (2017). Style transfer from non-parallel text by cross-alignment. In NIPS (pp. 6833-6844).
15. Stent, A., Marge, M., & Singhai, M. (2005). Evaluating evaluation methods for generation in the presence of variation. In CICLing (pp. 341-351).