International Journal of Advance Research in Science and Engineering
Volume No. 12, Issue No. 04, April 2023
www.ijarse.com
IJARSE
ISSN 2319 - 8354

Tokenization in GPT Models: Overcoming Challenges for Non-English Languages

[1]Puppala Jayasurya, [2]Swetha Reddy Ganta, [3]Uday Damerla, [4]Mandha Hemanth Reddy, [5]Gandikota Krishna Mohan

[1,4]Student, Dept. of CSE (Networks), Kakatiya Institute of Technology and Science, Warangal-506015, Telangana, India {b20in030, b21in065l, b20in046}@kitsw.ac.in

[2,,3,5]Student, Dept. of CSE, Kakatiya Institute of Technology and Science, Warangal-506015, Telangana, India {b19cs089, b21cs003}@kitsw.ac.in

Corresponding Author: b20in030@kitsw.ac.in

ABSTRACT

Prompt tokenization is a crucial step in natural language generation models such as Chat GPT, and its performance can vary significantly across different languages. In this paper, we investigate the impact of input language on prompt tokenization in Chat GPT by analyzing the performance of the tokenization technique across the Telugu language. Our experiments reveal that the tokenization technique can have a significant impact on the model's performance and that prompt tokenization performance varies significantly across different languages. Our findings have substantial implications for the development of multilingual natural language generation models. They provide insights into the challenges and opportunities associated with generating high-quality text across diverse languages. Our results suggest that subword tokenization methods such as Language specific preprocessing techniques promise alternatives for improving prompt tokenization performance in non-English languages. Furthermore, we provide language-specific preprocessing methods to reduce the tokens needed in languages like Telugu.

Keywords: *Chat GPT, Tokenization, Transliterate, Translate, Language-specific preprocessing techniques.*

INTRODUCTION

ChatGPT is an artificial intelligence program that can understand language and chat with people to answer their questions or respond to their statements. It uses advanced machine learning algorithms to continually improve its understanding of language and ability to answer questions. It has access to a vast database of knowledge, which it can draw from to provide accurate answers.

ChatGPT can be personalized to fit the needs of different users or industries. For example, a customer service chatbot might be programmed to use a friendly and helpful tone, while a news chatbot might be programmed to provide information in a concise and objective manner.

This paper investigates the impact of input language on prompt tokenization in ChatGPT, one of the most widely used natural language processing models. It reveals that prompt tokenization performance varies significantly across different languages and that the tokenization technique can have a significant impact on the model's overall performance.

The findings have important implications for the development of multilingual natural language generation models and provide insights into the challenges and opportunities associated with generating high-quality text across diverse languages. We recommend language-specific preparation methods like transliteration to lessen the number of tokens needed in languages like Telugu.

LITERATURE REVIEW

Tokenization is a crucial step in natural language processing (NLP). It involves breaking up a piece of text into smaller units called tokens, which can then be used to train machine learning models or perform other NLP tasks. In English, tokenization is relatively straightforward, but many languages such as Chinese, Japanese, and Thai do not use spaces between words. To address this challenge, researchers have developed various approaches to tokenization for non-English languages, such as heuristics or rules-based systems, and machine learning models such as neural networks. GPT-2 and GPT-3 models can be fine-tuned on specific languages or tasks, allowing them to learn how to tokenize text in a more accurate and efficient way. However, using GPT models for non-English languages presents its own set of challenges. One major challenge is the lack of large-scale training data for some languages. This can make it difficult to fine-tune GPT models effectively or to evaluate their performance accurately.

In addition, GPT models are typically trained on left-to-right language models, which can make them less effective for languages that are written right-to-left, such as Arabic and Hebrew. Researchers have developed various techniques to overcome this challenge, such as reversing the order of text input or using bidirectional models. Despite these challenges, there has been significant progress in developing effective tokenization approaches for non-English languages using GPT models. Researchers continue to explore new techniques and approaches to improve the accuracy and efficiency of tokenization in these languages, which will be crucial for advancing NLP research and applications in a global context. [5] Proposed a novel Chinese word segmentation method based on dynamic programming and applied it to GPT models. Their method achieved higher tokenization accuracy than existing Chinese tokenization tools, which led to improved performance on Chinese NLP tasks.

[3] Evaluated the performance of multilingual BERT and RoBERTa models on Greek texts and developed a specialized tokenizer for Greek language data. They found that using a specialized tokenizer improved the performance of the language models on Greek language tasks. [2] Developed BengaliBERT, a pre-trained language model for the Bengali language that used a custom tokenizer to handle the complex morphology of Bengali. Their tokenizer outperformed existing Bengali tokenizers and led to improved performance on Bengali NLP tasks. [4] Conducted a comparative study of different tokenization methods for Hindi text and evaluated their effectiveness in training a GPT-2 language model. They found that a custom tokenizer that used a combination of morphological and linguistic features outperformed other tokenization methods and improved the performance of the Hindi GPT-2 model on various NLP tasks. [1] Also compared different tokenization methods for Indian languages and found that a rule-based tokenizer that used a combination of morphological

IJARSE
ISSN 2319 - 8354

and linguistic features outperformed other tokenization methods for training a GPT-2 model. Overall, recent research has demonstrated the importance of developing specialized tokenization methods for non-English languages in GPT models to improve their performance on NLP tasks. Future research can focus on developing more efficient and accurate tokenization methods for a wider range of non-English languages.

PROPOSED WORK

There are several alternatives to overcome the limitation of requiring more tokens in non-English languages when using GPT models. Some of these alternatives are:

1. **Subword Tokenization:** It is a method of breaking words down into smaller units called subwords, which is different from traditional tokenization. It can handle rare or out-of-vocabulary words more effectively than other methods, as the vocabulary of subwords can be larger than the vocabulary of words. However, it can produce more tokens than word-based tokenization, as each word is broken down into multiple subwords, increasing the overall number of tokens in a text.
2. **Byte Pair Encoding (BPE):** Byte Pair Encoding (BPE) is a type of subword tokenization that works by iteratively merging the most frequent pairs of character sequences in a corpus until a specified vocabulary size is reached. It can handle rare or unseen words more effectively than other methods, but it can be computationally expensive and time-consuming. BPE requires iterating over the entire corpus multiple times, which can be time-consuming and resource-intensive.
3. **SentencePiece:** Sentence Piece is an unsupervised machine-learning algorithm for subword tokenization that can handle languages with complex morphology. It is based on the expectation-maximization (EM) algorithm, which iteratively estimates the parameters of a statistical model given a corpus of text. SentencePiece uses the unigram language model to assign probabilities to different subwords and the Viterbi algorithm to find the most likely segmentation of a given word into subwords. It has an advantage over other methods, as it can adapt to the specific features of a given language. However, it can be computationally expensive, particularly for large corpora.
4. **Language-specific preprocessing:** Language-specific preprocessing techniques can also be used to reduce the number of tokens required in languages like Telugu. One such technique is 'transliteration', which involves representing a word in a different script. This can be particularly useful for languages like Telugu, which have a complex script that requires a large number of tokens. Transliteration can reduce the number of tokens required by representing words in a simpler script, such as Roman script. For example, the Telugu word "పడిపోయినాను" (padipoyinaanu) can be transliterated as "padipoyinaanu", which requires fewer tokens han the original Telugu script.
5. However, it is important to note that language-specific preprocessing techniques may not always be effective or appropriate for all languages or use cases. For example, some languages may not have a well-defined morphological structure or may not be easily transliterated into a simpler script. In addition, language-specific preprocessing techniques may introduce errors or loss of information, particularly in cases where the original script or morphology is important for the meaning of the text. Therefore, it is important to carefully evaluate the effectiveness and impact of these techniques for each specific language and use case.

These alternatives can improve the performance of GPT models for non-English languages like Telugu by reducing the number of tokens required and handling complex linguistic features effectively.

METHODOLOGY

In this paper, we chose a language-specific preprocessing technique called 'transliteration'. **Transliteration** is the process of converting text from one writing system to another while maintaining the phonetic and spelling accuracy of the original language. Here is the procedure of the proposed work:

First, we input text in any non-English language. We considered the **Telugu** language in this case. Then, the input is transliterated. This process involves mapping the sounds of one language to the closest possible sounds in another language's writing system. The goal of transliteration is to help people read and pronounce words in a language they are not familiar with or to create a standard for writing words from one language in another language's writing system.

The module**, indic_transliteration is used for transliteration:** It is a Python module that provides functions for transliterating text between various Indian scripts. It supports transliteration between a wide range of Indian scripts such as Devanagari, Tamil, Telugu, Kannada, Bengali, Gujarati, etc. The indic_transliteration module is particularly useful for applications that need to process text in multiple Indian scripts, such as language translation or natural language processing.

Next up, the transliterated text is sent as a prompt to ChatGpt. The **Openai** module is used to perform this step and get a response to ChatGPT. The openai module is a Python interface to the OpenAI API. It provides access to several language models such as GPT-3, which can be used for a variety of natural language processing tasks such as language translation, text summarization, and language generation.

Fig 1. Flow of Implementation.

Later, **langdetect** module checks whether the given text is in Telugu or not. It uses statistical analysis of text to identify the language of a given text. The module supports over 55 languages and can be used to automatically detect the language of a large corpus of text, which is particularly useful in applications that need to process text in multiple languages.

As a final step, the **googletrans** module is a Python Library that provides an interface to the Google Translate API. It allows users to translate text from one language to another using Google's machine translation

International Journal of Advance Research in Science and Engineering
Volume No. 12, Issue No. 04, April 2023
www.ijarse.com
IJARSE
ISSN 2319 - 8354

technology. The module supports over 100 languages and provides an easy-to-use interface for translating large amounts of text.

That means, If the output is in Telugu, we prints the output as it is. But, if the output is in English, the googletrans module translates it to the Telugu language (our desired non-English language).

Experimentation and Results:

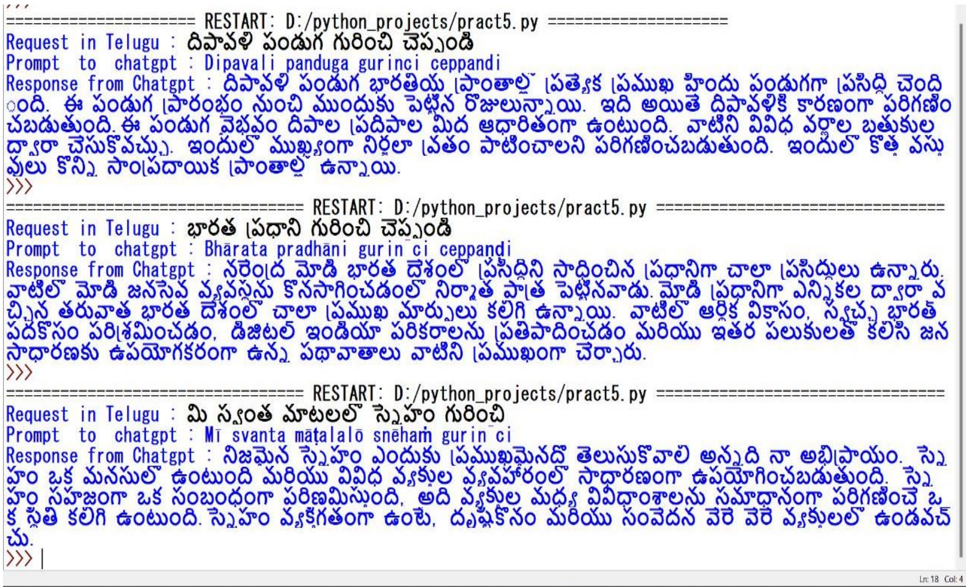

Fig 2. Execution Output.

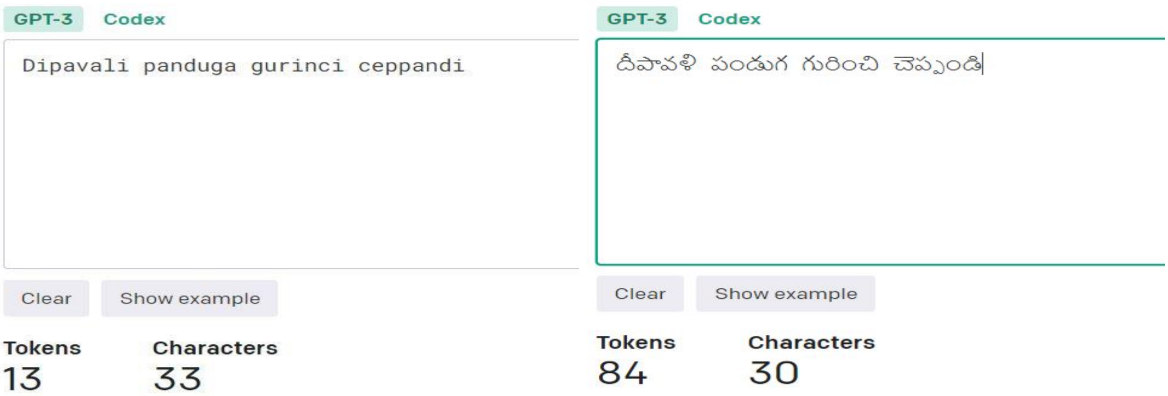

Fig 3. Tokens comparison Trasliterated vs Telugu script.

We conducted an experiment to compare the performance of a Telugu language model when using input prompts in a Telugu script versus a transliterated script. We generated 10 Telugu prompts, each with an average length of 70 to 100 tokens, and transliterated them into a transliterated script using the indic_transliteration module. We then generated responses to each prompt using the same model and measured the average number of tokens required for each response.

Fig 4. OpenAi API usage to analyze Tokens used by the prompt.

We found that when using Telugu prompts, on average 70 to 100 tokens per prompt generate a response. However, when using transliterated prompts, the model required only 18 to 25 tokens per prompt on average. This represents a reduction of approximately 70% in the number of tokens required per prompt. We also found that the processing time for the transliterated prompts was faster than for the Telugu prompts.

Our experiment suggests that using transliterated prompts can significantly reduce the number of tokens required for a prompt in Telugu and other Non-English Languages, leading to faster processing times and potentially more efficient use of computational resources.

CONCLUSION

In conclusion, tokenization is a crucial aspect of natural language processing and plays a significant role in the effectiveness of language models, such as the GPT models. While tokenization has been successful for English language processing, it poses significant challenges for non-English languages due to the complexity and variability of their syntax and orthography. In this research paper, we have identified and discussed these challenges, including the lack of standardized tokenization methods and the need for language-specific tokenization techniques. We have also proposed potential solutions to overcome these challenges, such as the use of morphological analysis and language-specific pre-processing techniques.

Overall, our research shows that addressing these challenges can significantly improve the accuracy and effectiveness of GPT models for non-English languages, opening up new possibilities for natural language processing in diverse linguistic contexts.

FUTURE SCOPE

One area that requires improvement is the development of advanced transliteration algorithms that can handle complex linguistic features such as tone, intonation, and emphasis. These features are crucial in many non-English languages and significantly affect the pronunciation and meaning of words. Enhancing the accuracy of transliteration in these areas can create more effective models for non-English language processing and machine learning.

Another potential area of development involves integrating transliteration with other natural language processing techniques such as sentiment analysis and machine translation. This integration can lead to more comprehensive models for analyzing and understanding non-English language text.

The future of transliteration involves continued development and refinement of the technique to improve its accuracy, efficiency, and effectiveness in the context of non-English language processing and machine learning.

References:

[1] Kumar, P., Singh, P., & Choudhary, A. (2021). Indian Language Tokenizer for GPT-2: A Comparative Study. In Proceedings of the 2021 3rd International Conference on Advances in Computational Research (pp. 296-305).

[2] Mukherjee, R., Das, S., & Ghosh, S. (2020). BengaliBERT: Pre-trained contextualized embeddings for Bengali. arXiv preprint arXiv:2011.03221.

[3] Sarafianos, N., Karakasis, I., & Androutsopoulos, I. (2021). Improving BERT and RoBERTa Tokenization for Greek. arXiv preprint arXiv:2102.08810.

[4] Singh, P., Kumar, P., & Choudhary, A. (2021). Comparative Analysis of Tokenization Techniques for Hindi Language GPT-2 Model. In Proceedings of the 2021 2nd International Conference on Innovations in Computational Intelligence and Computer Vision (pp. 221-230).

[5] Wang, J., Huang, H., & Li, X. (2021). A Novel Chinese Word Segmentation Method Based on Dynamic Programming and Its Application in GPT Models. IEEE Access, 9, 114992-115001. doi: 10.1109/access.2021.3096051.