A COMPLEXITY-AWARE WEB SEARCHING PARADIGM TO IMPROVE USER PRODUCTIVITY USING NATURAL LANGUAGE PROCESSING AND THE DISTILBERT TRANSFORMER MODEL

Ruiyi Zhang[1], Carlos Gonzalez[2]

[1]Troy High School, 2200 Dorothy Ln, Fullerton, CA 92831
[2]Computer Science Department, California State Polytechnic University, Pomona, CA 91768

ABSTRACT

Search engines (Google search, Bing search, etc.) have had great success over the past decade, promoting productivity in almost every area. Based on user inputs, search engines are able to present users with lists of related contents (links) and previews. More recently, high-level human-like responses combining various searched contents are being made possible due to recent advancements in large language models (LLM). However, oftentimes, users still find it still hard to quickly navigate to the contents they really look for and demand a better searching framework. For example, some users might waste time skimming through lots of technical details when they just hope to have an overview. We examine this user demand and believe a complexity-aware pipeline could greatly help with this inconvenience. More specifically, we propose a searching paradigm that analyzes results from standard search engines by their complexities first, and then present users with complexity-labeled contents through a new user interface design. Through this new searching paradigm, we aim to present users with more customized search results sorted by their complexity labels with consideration to user intent, whether that would be a high-level overview or a detailed technical inspection. This is done through utilizing state-of-the-art transformer models fine-tuned on our custom-made dataset and modified for our intent.

KEYWORDS

Transformer, Natural Language Processing, Complexity-Aware, Web Search

1. INTRODUCTION

Search engines (Google search, Bing search, etc.) have had great success over the past decade, promoting productivity in almost every area. Based on user inputs, search engines are able to present users with lists of related contents (links) and previews. More recently, high-level human-like responses combining various searched contents are being made possible due to recent advancements in large language models (LLM) [13]. However, oftentimes, users still find it still hard to quickly navigate to the contents they really look for. For example, some users might waste time skimming through lots of technical details when they just hope to have an overview. Additionally, content returned by relevant search engines are often highly manipulated by practices such as link-building and Search Engine Optimization (SEO), contributing to users spending more time encountering irrelevant content. In fact, statistics show that “74.3% of link builders pay for links” and “66.5% of links to sites over the last nine years are dead”[1]. This

David C. Wyld et al. (Eds): AdNLP, CDKP, NCO, SAI, MLT, Signal, SOFT, ICAITA, CMC - 2024
pp. 31-42, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.141004

exaggerates the problem where the top results of Google captures 27.6% of all clicks while pages beyond the first only capture 0.63% of all clicks, suggesting that certain quality content lacking the support of SEO manipulations are being obscured from most users [2]. Therefore, internet users ranging from power researchers to daily users alike demand an improved framework to address these shortcomings and enhance their productivity.

The methodology comparison within the paper highlights the evolution of text complexity analysis and the role of AI in enhancing user search experiences.

Methodology A emphasized the breakthrough of Google's Search Generative Experience. This demonstrates the utilization of cutting-edge technology such as the large language model and Generative Pre-Trained Transformer. This technology is able to generate the users with a direct and surface level answer to their query, with citations to the source of information. However, it fails to cater to specific user needs for complexity-aware and direct access sorted results.

Methodology B presented the Coleman-Liau Index, a formulaic readability test that evaluates text complexity with solely numerical considerations. This index completely disregards any semantic or contextual language layer comprehension. This obscures much of the text complexity as well as the level of knowledge one can gain from the text, rendering this index completely impractical for the goal of sorting webpages for user productivity and convenience.

Methodology C involved the Latent Semantic Analysis method, which is powerful for extracting word-context relationships. However, it is unable to capture the multidimensional language patterns as efficiently as our proposed AI-driven paradigm due to its static nature and demand for substantial computational resources, especially for a corpus the size of the internet.

Together, these comparisons depict the advancements of our proposed model over many traditional methods, demonstrating its ability to be the cutting-edge product to fulfill the niche of users in search of complexity-sorted search results.

We examine this user demand and believe a complexity-aware pipeline could significantly alleviate this inconvenience. More specifically, we propose a searching paradigm that analyzes results from standard search engines by their complexities first with a state-of-the-art transformer model that is fine-tuned on a custom-made dataset, and then presents users with complexity-labeled contents through a new user interface design. The utilization of a transformer model is efficacious on both a practical level and a theoretical level as its self-attention mechanism enables it to weigh the importance of different words within an input and capture the nuances of language, a cardinal pillar of complexity analysis [3]. Additionally, the practical successes of transformer models have been evidenced by Generative Pre-trained Transformer (GPT) and models like Bidirectional Encoder Representations from Transformers(BERT), both demonstrating their superiority in context comprehension and human-like text generation. The custom-made dataset is to be created jointly with our multi-threaded web scraping algorithm for content obtainment and GPT-4 for content complexity analysis. Through this new searching paradigm, we aim to present users with more customized search results sorted by their complexity labels with consideration to user intent, whether that would be a high-level overview or a detailed technical inspection.

Experiment A rigorously tested our model for websites under 400 words which exceed the capabilities of our tri-section sampling methodology. The results were generally promising, with only a mere 1.63724% minimum divergence from the standard validation loss. However, there was an increase in divergence as the website word count dropped below 324. However, this is expected as informative websites under 324 words are relatively rare. As a whole, these results attests to the model's strong capability in managing outliers effectively without compromise to the user experience.

Experiment B expanded our scope, examining the model's adaptability across diverse content genres. The experiment encompassed texts highly reflective of real-world user scenarios from politics, fine arts, fiction, blog posts, and food. The analysis demonstrated a miniscule and consistent increase in loss across unfamiliar genres, with particular spikes in food and fiction genres. Although unlikely to negatively impact user experience, the insights from this experiment guides our efforts to refine the model with more genre-specific data to better ensure user satisfaction.
These two experiments testifies to our model's robustness in various circumstances that a user may encounter. Although it raised indicators to areas of improvements, it has provided a clear direction for future enhancements that will further serve the nuanced needs of every user.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Avoiding Automated Web Scraping Prohibitions

In creating a pipeline that can handle parallel requests to the web scraping algorithm, avoiding automated web scraping prohibitions are crucial. For many websites, after more than two scrapes from the identical IP Address, it will prompt a mandatory reCAPTCHA. Hence, we understood that it is crucial that we develop a rotating proxy system that counteracts this issue. We believe that we could compile various arrays of available proxies that are assigned to unique users of our application. Additionally, to ensure a cohesive experience for our users, browsing speed is a necessity. Hence, we could develop an advanced multithreaded algorithm to utilize the full extent of the cloud computer.

2.2. Gathering a Robust Dataset

Gathering a robust dataset is cardinal in ensuring the full functionality of our pipeline. However, there is no existing dataset with a large variety of websites labeled with a complexity score that fits the criterias which we value. Therefore, to generate a dataset with at least five million words and twenty-thousand websites each labeled with a complexity score that is meticulously generated, we could first use our advanced web scraping algorithm and take advantage of the rising generative AI applications to label our dataset with complexity scores. With careful and extensive prompt engineering and testing, a prompt can be made so that GPT-4 can objectively, transparently, and logically generate complexity labels. Additionally, with consideration to the price of GPT-4 API tokens, it is crucial that the creation of our dataset avoids websites that have been poorly scraped. Hence, we believe that the incorporation of RoBERTa, a model built upon the same architecture of BERT that is highly capable of text classification, can help detect and remove failed scrapes before GPT-4 complexity label processing [4].

2.3. The Maximum Number of Tokens

One of the intrinsic limitations of contemporary transformer models, including DistilBERT, is their restrictions on the maximum number of tokens that can be processed in a single input sequence. Specifically, DistilBERT is constrained to 512 maximum input tokens[5]. This limitation is rooted in the model's innate architecture and the self-attention mechanism's time complexity, which scales quadratically with the length of the input sentence. Given the importance of context in complexity scoring for longer website articles, this token limit poses a significant challenge to this pipeline. Hence, in order to maintain the high accuracy and fast performance of DistilBERT, we could potentially implement a sliding window techniquethat moves a predetermined fixed-size window across the text where each capture overlaps the previous [6]. This overlap between windows can help preserve context. However, this method is relatively computationally inefficient. Hence, we plan to experiment with our own tri-section sampling technique to address this challenge with both computational efficiency and result quality in consideration.

3. SOLUTION

This program is built upon three fundamental pillars: a web content retrieval and processing system, a custom trained content analysis transformer model, and an API service that enables deployment on various means. When a query request is made from the user interface, our pipeline will first retrieve the web content of the top 50 results from google and process the raw HTML data into coherent and interpretable content. Various essential elements of content, such as original URL, domain, and title, are organized and preserved into .csv files readable by the pandas library. To ensure efficiency in our application as well as meet the maximum token expectations of the model architectures used in our program, we use our unique Tri-Section Sampling method to truncate the content of each website to only the first 50 words, middle 300 word, and last 50 words. This data is then passed through the RoBERTa Model trained on the Corpus of Linguistic Acceptability (CoLA) dataset to ensure minimum data quality [7]. This concludes the first major component of data retrieval and processing. This data is then passed into our complexity analysis DistilBERT model and a score is assigned for each website. Each label is appended to the database, and this concludes our second major component. Then, the data is processed by our ranking algorithm and the websites are sorted from simplest to the most complex. Ultimately, this information is sent by our cloud server as a JSON file to the frontend and displayed to the users in a highly interpretable and transparent manner, with each of the complexity rating scores displayed.

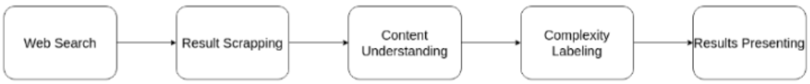


Figure 1. Design overview

The most important element of our web scraping algorithm is its efficient and effective data preprocessing abilities. Our system utilizes Regular Expressions (RegEx) and Natural Language Toolkit's (NLTK) stopwords dataset to eliminate noise in the raw HTML data. This algorithm is also capable of removing non-body hyperlink elements to further improve model performance and text interpretation.

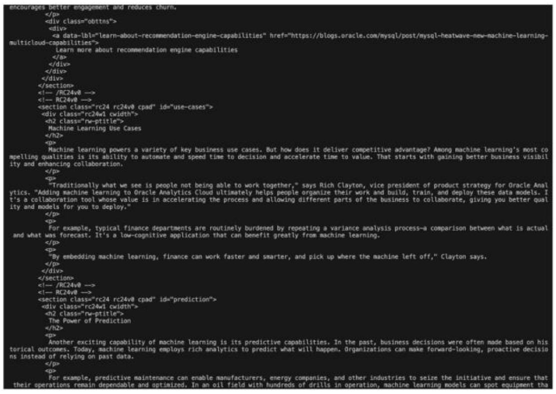


Figure 2. Raw HTML

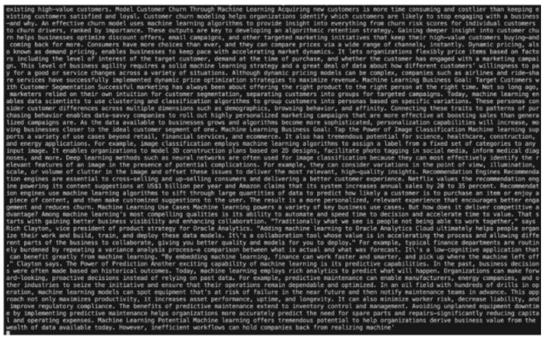
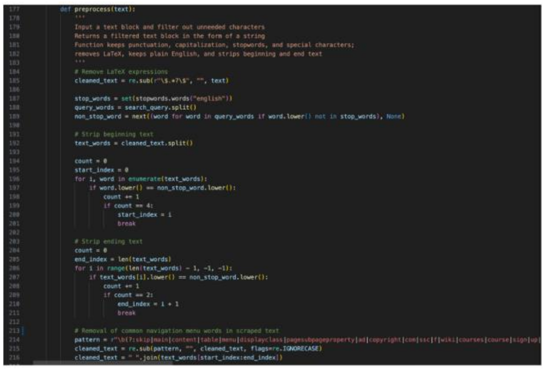


Figure 3. Preprocessed data



```
def preprocess(text):
    '''
    Input a text block and filter out unneeded characters
    Returns a filtered text block in the form of a string
    Function keeps punctuation, capitalization, stopwords, and special characters;
    removes LaTeX, keeps plain English, and strips beginning and end text
    '''
    # Remove LaTeX expressions
    cleaned_text = re.sub(r"\$.+?\$", "", text)

    stop_words = set(stopwords.words("english"))
    query_words = search_query.split()
    non_stop_word = next((word for word in query_words if word.lower() not in stop_words), None)

    # Strip beginning text
    text_words = cleaned_text.split()

    count = 0
    start_index = 0
    for i, word in enumerate(text_words):
        if word.lower() == non_stop_word.lower():
            count += 1
            if count == 4:
                start_index = i
                break

    # Strip ending text
    count = 0
    end_index = len(text_words)
    for i in range(len(text_words) - 1, -1, -1):
        if text_words[i].lower() == non_stop_word.lower():
            count += 1
            if count == 2:
                end_index = i + 1
                break

    # Removal of common navigation menu words in scraped text
    pattern = r"\b(?:skip|main|content|table|menu|displayclass|pagesubpageproperty|ad|copyright|com|ssc|f|wiki|courses|course|sign|up|
    cleaned_text = re.sub(pattern, "", cleaned_text, flags=re.IGNORECASE)
    cleaned_text = " ".join(text_words[start_index:end_index])
```

Figure 4. Screenshot of code 1

The method "preprocess" best represents this component. The purpose of this method is to thoroughly eliminate interference and noise within the scraped raw HTML from each website of the query. This method is called for every website that is returned by the web scrape. When "preprocess" is called by the pipeline, it will first remove all LaTex expressions on a given website HTML by using regular expressions to identify the backslash, which is an indicator for LaTex. The "re.sub" expression used here essentially finds the LaTex expression and substitutes it for an empty string. This is particularly important as our pipeline is often applied to academic research, often involving websites with many LaTex expressions. Upon the removal of the LaTex expressions, the "preprocess" method references the NLTK library's dataset which contains a set of common English stop words--words such as "and", "the", and "is" that provide little to no contribution to the comprehension of a text. Hence, removing them will accelerate the pipeline's efficiency and increase the quality of text used to train and evaluate the model. After the initial cleanup, this method aims to identify where the meaningful content begins and ends. To do this, the method searches through the list for the first key term from the user's search query that is not a common stop word. The process notes the positions of these key terms, and upon finding the fourth instance, it sets this position as the start for the content-rich section. This allows the algorithm to skip over the introductory text, often containing generic and irrelevant information. The same is done to the end of the content. Lastly, we aim to remove any navigation menu words that have persisted. This was done through the reference of a bank of common navigation menu words and using RegEx to remove them. Following these steps, the text is ready to be analyzed by the model with high accuracy and effectiveness.

General Overview:

The Artificial Intelligence component that directs the complexity rating assignment harnesses the power of the DistilBERT architecture. Originating from the BERT model, DistilBERT captures the revolutionary core language understanding capabilities of BERT with a more streamlined and efficient design. DistilBERT has 40% less parameters than BERT and is able to preserve 97% of BERT's performances while achieving a 60% increase in time efficiency. [5]. With a focus on efficiency and practicality, DistilBERT was the optimal model for our central Artificial Intelligence component.

Specific Method explanation:

Our use of assigning complexity scores to text is relatively specific and nuanced for transformer models, which provides us with no pre-existing datasets that meet our expectations. Therefore, we created a dataset to accommodate this challenge. The creation of this dataset first involved scraping over 5 million words from 20,000 thousand unique websites to generate a sufficient quantity of content for our model training. However, the core of our data generation relied on the recent LLM and GPT breakthroughs by engineering a prompt that is capable of utilizing GPT-4 API to generate complexity scores from each website transparently, objectively, and consistently. Below were our criteria for complexity rating.

- The frequency and complexity of specialized terminologies.
- The sophistication and profundity of conveyed technical notions.
- The cognitive requirement essential to grasp the subject matter.
- The length of the website content, indicating the scope of the content.
- The projected target audience of the website.

We also implemented the approach of Explainable AI (XAI) to ensure that the Large Language Model can self-inspect and improve its conclusions. The data set generation and training of our model adhered to our tri-section sampling method where only the first 50 words, middle 300 words, and last 50 words were processed and considered. This model is trained with the state-of-the-art Adam Optimizer and Mean Square Error Loss Function given the quantitative nature of our model's output [8]. The training lasted 10 epochs with a batch size of 64, ensuring the model's capability to comprehend the nuances that exist within language while preventing overfitting.

General Overview: The final major component to this pipeline is the integration to Google Cloud where it is accessible as an API [12]. This component uses "flask" and to enable Google Cloud to run a python script that uses the pipeline to process the user's requests. Its accessibility as an API enables the deployment of this pipeline onto websites, such as our own--https://www.searchsense.us.

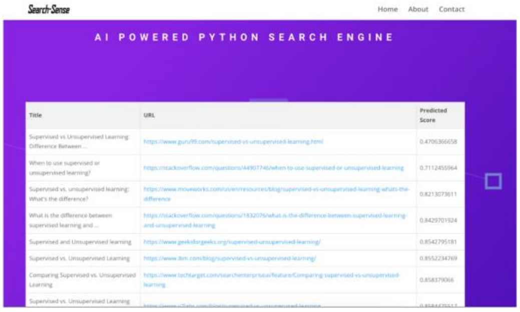


Figure 5. Screenshot of UI

```python
def get_result(search_query):
    def make_prediction(text, model, tokenizer):

        # Convert output to label
        predicted_score = output.item()
        return predicted_score

    df['predicted_scores'] = df['Text'].apply(lambda x: make_prediction(x, model, tokenizer))
    df = df.sort_values(by = ["predicted_scores"])

    for index, row in df.iterrows():
        title = row['Title']
        url = row['URL']
        predicted_score = row['predicted_scores']

        if predicted_score < 0.5:
            difficulty = "Low"
        elif 0.5 <= predicted_score <= 0.88:
            difficulty = "Moderate"
        else:
            difficulty = "High"

        print(f"""
        Title: {title}
        URL: {url}
        Difficulty: {difficulty} Model Score: ({predicted_score})
        ----------------------------------------------------------
        """)

    return df.to_json(orient = "records")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Extract search query from request
        search_query = request.json["search_query"]
        # Use your get_result function with the provided search_query
        results = get_result(search_query)
        return jsonify(results)
    except Exception as e:
        return jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8080, debug=True)
```

Figure 6. Screenshot of code 2

Specific Explanation: This code sample best represents the capability of Flask and the data that is sent from the API to the receiving service. In the flask application, a function named “get_result” is defined and this function serves as the actuator of the entire pipeline. The “search_query” which the “get_result” function demands is extracted from the request made from the user interface. As soon as the “make_prediction” function is called inside of “get_result”, the array of “predicted_scores” is applied to the dataframe as a new column. The transformed DataFrame now enables our pipeline to analyze and assign different difficulty levels and sort the results in a manner that best fits the user’s interest and preferences. Any scores below 0.5 is determined as low difficulty, any scores between 0.5 and 0.88 is determined as moderate, and any scores higher than 0.88 is determined as high difficulty. We decided these thresholds through repeated analyzing data from various search queries on diverse subjects to ensure that the users will be greeted by a consistent number of low, medium, and high level of difficulty results. These thresholds were further authenticated and adjusted by empirical analysis to ensure that results of low, medium, or high difficulty generally align appropriately within the expected parameters of complexity.

4. EXPERIMENT

4.1. Experiment 1

The nature of our tri-section sampling methodology demands a minimum of 400 words from a website for optimal performance in delineating content complexity. However, there exist many websites that fail to fulfill the criteria of the 400-word minimum. Hence, an experiment has been conducted to ensure low discrepancy between the normal validation loss and the loss for websites below our threshold. It should be noted that during training, the model did confront websites that did not satisfy the 400-word minimum requirement.

In the design of this experiment, attention was centered upon the variation of discrepancy levels in respect to the deviations in word count from the established 400-word benchmark. Therefore, 400 websites with total word counts ranging from 250 words to 400 words were gathered and analyzed by both GPT-4 in the same manner as the dataset generation and our DistilBERT model. Through this, we created 2-Dimensional tuples in the format of "(word count, loss)" to be analyzed graphically and mathematically. This approach enables us to observe the effects of varying word count on the model's loss output, providing insight into the robustness of our tri-section sampling methodology. The selected range of word counts ensures diversity while continuing to reflect the realistic scenarios that our users may encounter. Hence, the findings of this experiment will be both relevant and applicable to real-world purposes.

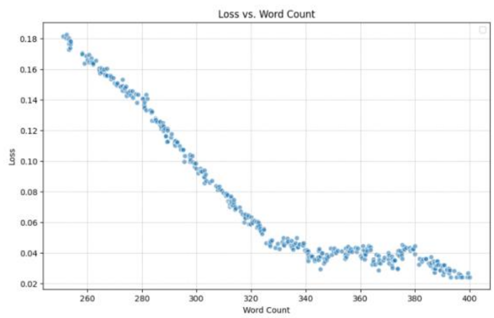


Figure 7. Figure of experiment 1

The trend displayed by this graphical visualization is consistent with the theoretical predictions as the discrepancy decreases as the word count converges with the benchmark of 400 words. Specifically, the minimum loss of this experiment only diverges 1.63724% from the validation loss of 0.025286186300218106. The analysis of the data reveals an average loss of approximately 0.0752, a 197.55% increase from the validation loss. Though this percent increase appears alarming at first sight, it attests to the model's success in achieving an exceptionally low loss and demonstrates a realistic and expected discrepancy that affects the final user experience minimally. However, we were intrigued to find that the loss increases in a more drastic manner after decreasing below 324 words. This suggests that during training, our dataset contained a small quantity of websites below 324 words. However, this was expected because informative websites below 324 words occur rarely on the internet and in our data collection process. This experiment has assured that the complexity rating model is capable of successfully handling outlier websites that could have interfered negatively with the user experience.

4.2. Experiment 2

Throughout the creation of our pipeline, we have placed an emphasis on STEM related queries as our audience pertains most closely to that of the research or academic community. However, this pipeline is also curated for daily use, ranging from politics to fine arts. Therefore, the second experiment is destined to evaluate the performance and accuracy of our model across various genres from niche to mainstream.

For the design of the second experiment, a comprehensive dataset has been curated, comprising 500 websites across 5 diverse genres: politics, fine arts, fiction, blog posts, and food. These categories were deliberately chosen to span a large range of genres that a daily user may frequently encounter. Each category contributes a distinct textual structure, complexity, and stylistic elements, each of which are crucial in evaluating the model's adaptability and accuracy. Each sample of data in this dataset has undergone the same preprocessing, tri-section sampling methodologies, and GPT-4 evaluation as the training and validation dataset. The protocols for analysis will focus on comparing the performance metrics across these 5 distinct genres. The model's performance and predictions will also be qualitatively assessed by humans as well as GPT-4 to ensure consistency via cross-authentication.

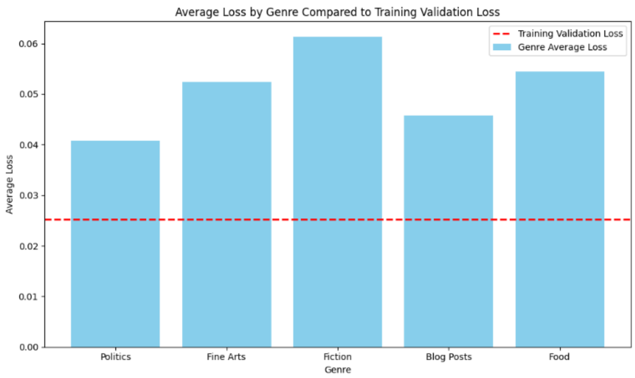


Figure 8. Figure of experiment 2

The results from this experiment reveal various noteworthy aspects about the model's performance with unfamiliar text types and content categories. The bar chart, which compares the average loss values across five different genres relative to the established benchmark loss from the training validation dataset, demonstrates a holistic increase in loss. This aligns with our expectations and affairs our confidence in the model's robustness and generalization capabilities. This increment in loss is not anticipated to impact the user experience or compromise on the precision of the sorting algorithm. Additionally, it appears that the model exhibits particular challenges when confronting content from the food and fiction genres. This suggests the nuances in these genres--likely the sensory-laden language of food content and the imaginative and diverse narrative styles found in fiction--that are less compliant to the model's current state of training. In order to address this, we plan on training the model further with more data from these two particular genres.

5. Related Work

In recent years, Artificial Intelligence (AI) has observed tremendous development and popularity. Due to this, many are seeking the implementation of Artificial Intelligence in web searching in hopes of increasing user productivity and convenience. In fact, Google has recently achieved a breakthrough in this field with the Search Generative Experience (SGE). Google SGE strives to simplify the user experience by providing users directly with an overview of the answer to their query through generative AI with reference to the source of information [9]. Additionally, they have implemented BERT into the query comprehension process to further enhance their results. This methodology is successful at increasing user productivity by rapidly returning a general answer to the query prompted. However, it often fails to provide the user with a website that fits their level of intended complexity. This is attributed to the fundamental nature of SGE as the website it selects and cites for its AI generated response disregards the user's intended level of complexity. Hence, if a user wishes to dive deeper into the topic, this user is nevertheless faced with the inconvenience of exploring countless websites to accomplish their goal. Our paradigm, however, is capable of understanding a user's interests and returns the results sorted by their level of complexity, thus alleviating the need for them to explore endlessly through search results to locate the content that matches their expertise or interest level.

Despite the popularity in AI, there exist various approaches that rely on a formulaic methodology for machine text complexity scoring. One prominent method is the Coleman-Liau Index[10]. This methodology calculates text complexity using letters per 100 words and sentences per 100 words, bypassing the need for syllable estimation, which drastically increases the efficiency for computerized scoring. However, the Coleman-Liau Index, like other formulaic readability tests, does not account for the semantic or contextual layers of language that often drastically influence the complexity level of a text. Rather, it simplifies the text to a numerical level, disregarding the nuances like the vocabulary, jargons, or depth of content explored. Each of these criteria is essential for the determination of the true readability from a human perspective. In contrast, our AI-driven paradigm expands upon this by leveraging contextual and semantic processing capabilities with state-of-the-art transformer models. It accounts for not only the structural elements of language, but also its inherent meaning and the knowledge base the text implies for readers. Therefore, our pipeline can offer the users with more tailored and curated results that precisely fits their expectations.

As we continue to explore the landscape of text complexity analysis, it becomes imperative to examine the contributions of neural network-based methods. These approaches leverage deep learning techniques for text comprehension and prediction, foundationally similar to that of our transformer model. One powerful method which scientists have explored is Latent Semantic Analysis (LSA), a technique that uncovers obscured relationships between words within large sets of text by employing singular value decomposition [11]. This technique delves into the linguistic attributes of texts, evaluating beyond mere surface-level indicators through focusing on the patterns of word distributions to capture the multidimensionality of language. Throughout its existence, it has earned the recognition of experts in the field of linguistics. However, there are various limitations that restricts it from being practical in flexible situations. One major limitation is that LSA requires substantial computational resources for processing, especially if the corpus size is large. Therefore, given the size of the internet and the subgenres of text within it, LSA is highly inapplicable towards this ambition. Additionally, LSA is static in nature; once the analysis is performed, it does not learn or adapt from new data unless the model's restrained corpus is updated. However, our pipeline overcomes these limitations with its strong generalization abilities as well as its light-weight and efficient processing speeds. Therefore, the model demands exposure from only a miniscule segment of subspaces of the internet for it to gain its ability to understand and rank texts on their complexities. Therefore, our pipeline is easily expandable and deployable for various practical circumstances.

6. CONCLUSIONS

This paradigm, though functional and practical as of its current stage, does contain certain limitations that impede its ability to be universally applicable across all scenarios and audiences. One particular limitation is its overgeneralization of a wide range of text categories with only a singular model. We were fortunate that our model was able to reach a compromise between its generalization abilities and its high accuracy. However, to be able to deliver our pipeline on an industrial scale to the public, we plan on training genre specific models and enabling our pipeline to select the appropriate model for different genres of text. This drastically decreases the chances of users being faced with incorrect rankings that may negatively impact their experience. Additionally, the nature of our web scraping algorithm is dependent on network capabilities. As tested previously, the entire pipeline is able to execute in as little as 2.3 seconds with a 500 Mbps download bandwidth. Therefore, we intend to enhance collaboration with Google Cloud to secure an expansion of our network bandwidth to streamline the user experience and productivity. Lastly, we committed to improving user privacy with our API requests and returns as this is most essential for a large-scale deployment to the general public.

This research has embarked on an exploration into enhancing user productivity and convenience through a complexity-aware web searching paradigm. Our comparative analysis reveals a shift from traditional formulaic approaches to the more nuanced AI-driven models, with emphasis on the model's ability to tailor search results to the user's individual complexity preferences [14]. Over the course development, we have demonstrated the model's effectiveness in content complexity scoring. As we advance, we are committed to further refinements of our model to enhance its capability in the broad and constantly expanding internet landscape. This work serves to advance the academic discussions on the role of AI in web searching technologies as well as laying the foundation for future innovations on improving the interface between digital information and human comprehension [15].

REFERENCES

[1] Kaminski, Mikolaj, Wojciech Marlicz, and Anastasios Koulaouzidis. "Googling on colonoscopy: a retrospective analysis of search engine statistics." Clinical and Experimental Gastroenterology (2020): 397-405.

[2] Volokh, Eugene, and Donald M. Falk. "Google: First amendment protection for search engine search results." JL Econ. & Pol'y 8 (2011): 883.

[3] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[4] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[5] Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv preprint arXiv:1910.01108 (2019).

[6] Zhu, Chen, et al. "Long-short transformer: Efficient transformers for language and vision." Advances in neural information processing systems 34 (2021): 17723-17736.

[7] Warstadt, Alex, Amanpreet Singh, and Samuel R. Bowman. "Neural network acceptability judgments." Transactions of the Association for Computational Linguistics 7 (2019): 625-641.

[8] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[9] Frey, Carl Benedikt, and Michael Osborne. "Generative AI and the future of work: a reappraisal." Brown Journal of World Affairs 30.1 (2023).

[10] Coleman, Meri, and Ta Lin Liau. "A computer readability formula designed for machine scoring." Journal of Applied Psychology 60.2 (1975): 283.

[11] Korytár, Peter, et al. "Practical fast gas chromatography: methods, instrumentation and applications." TrAC Trends in Analytical Chemistry 21.9-10 (2002): 558-572.

[12] Bisong, Ekaba, and Ekaba Bisong. "An overview of google cloud platform services." Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners (2019): 7-10.
[13] Kasneci, Enkelejda, et al. "ChatGPT for good? On opportunities and challenges of large language models for education." Learning and individual differences 103 (2023): 102274.
[14] Cau, Riccardo, et al. "Long-COVID diagnosis: From diagnostic to advanced AI-driven models." European journal of radiology 148 (2022): 110164.
[15] Hu, Wen-Chen, et al. "An overview of world wide web search technologies." Proceedings of the fifth world multi conference on system, cybernetics and informatics. 2001.