The readability and similarity of ChatGPT with respect to Stack Overflow

Alan McKay, Hayden Westphal, Jordyn Iannuzzelli, Nathan Hine
University of Iowa, Group 7
{amckay, hawestphal, jiannuzzelli, nhine}@uiowa.edu

1. INTRODUCTION

With the rapid development of artificial intelligence, language models such as ChatGPT have made considerable progress in generating human-like responses to prompts. However, it is unclear how well these language models perform compared to other humans, particularly in terms of readability. Readability is the ease with which a text can be understood by a reader, and is influenced by factors like sentence length, word choice, and overall coherence.

Stack Overflow is a popular online community where developers can ask and answer technical questions related to programming. However, the vast amount of information on Stack Overflow can be overwhelming and time-consuming to navigate. To address this issue, language models such as ChatGPT are leveraged to generate answers to Stack Overflow prompts. However, it remains unclear how well these language models perform compared to the answers provided by human supplied answers on Stack Overflow.

This research paper aims to compare the quality and similarity of answers generated by ChatGPT with respect to human participants on Stack Overflow. Specifically, we will analyze the readability and similarity of answers provided by ChatGPT and compare them to the answers provided by accounts on Stack Overflow for the same prompt.

To achieve this goal, we will select a set of Stack Overflow prompts and their top-rated answers by the users. We will then use ChatGPT to generate answers to the same prompts and compare the readability and gauge similarity.

This research is important because it will display the strengths and limitations of language models such as ChatGPT and provide insight into how they may be improved. Furthermore, the results of this study may have implications for the development of language models in areas such as education, where similarity and readability are key factors in assessing the effectiveness of educational materials.

In this research paper, we aim to answer several questions related to ChatGPT's responses to programming-related questions. Specifically, we want to compare the similarity between ChatGPT's responses and those found on Stack Overflow, a popular online community for programming questions and answers. By addressing these questions, we hope to provide insights into the similarity and readability of ChatGPT's responses, as well as its potential use as a tool for technical support and education in the field of programming.

Our analysis has revealed several important findings related to ChatGPT's responses to programming-related questions. Firstly, we found that ChatGPT's responses have similar readability scores to higher rated Stack Overflow responses, indicating that ChatGPT can generate responses that are easy to read. However, we also found that Stack Overflow responses tend to be slightly easier to read than ChatGPT's. Secondly, our analysis showed that ChatGPT's responses are highly alike to the Stack Overflow responses of higher post ratings, indicating that ChatGPT has learned programming concepts and can generate responses that are similar. Overall, our findings suggest that ChatGPT has potential as a tool for technical support and education in the field of programming, and further research is needed to improve the readability of its responses.

2. RELATED RESEARCH

The paper, 'Learning to Program Using Online Forums' (Lau and Mitrovic) [5] shows that Stack Overflow is a common first resource when it comes to discussing code solutions. This is discussed in the context of Reddit, one of the largest internet forums. Intuit would dictate the diffusion of content from Stack Overflow for its ease of access and that its content would be diffused beyond Reddit. This provides a strong basis of comparison to potential AI generated answers.

Similarity for this study is evaluated with respect to answers supplied on Stack Overflow. A study by Chen et al. [2018] discusses behavioral patterns of answerers on the platform. Specifically, a look at correlation of complexity of answer, and frequency an answerer provides an answer, the time taken for an answer to be provided, etc. It finds that non-frequent writers of solutions tend to more complex problems – problems which take more time to have a solution provided. The complement is also noted, where 86-96% of accepted answers are written by frequent answerers. This suggests a possible bias in terms of these two types of answers in which ChatGPT, a single entity, may not adhere to- more discussion on this in the limitations section below.

Not only does this study look at textual measures of similarity, but also considers code similarity. A paper by Schleimer et al. [2003] provides a formal framework in which code similarity can be measured. The application of this framework is given by Stanford's MOSS web application.[6]

3. METHODS

Information was taken from Stack Overflow and ChatGPT, including questions, answers, and dates. After collection, the data was run through a series of methods for analysis, such as cosine similarity, Dale Chall readability scoring, and Flesch reading ease scores. The text below outlines the retrieval processes for the data in both Stack Overflow and ChatGPT, as well as the explanation for each analyzation method and calculation.

Section 3.1 – Gathering Stack Overflow Data

The first phase of data collection included gathering questions and answers from users on Stack Overflow. To ensure there were not any ChatGPT generated answers, the data collected was from prior to 2019 (ChatGPT-3's official release date; it should be noted that the program was given public access in 2022). All the data that was scraped from Stack Overflow was obtained in posts through a public API. The questions were selected from the categorization tags Python. Java, and JavaScript. 1100 top-rated questions were selected in each of these categories. In addition to textual information, a responses' date was maintained in the collection for temporal analysis and verification. Furthermore, an answer gathered from Stack Overflow had its associated post-rating maintained within the same dataset. The data was then compiled and stored as JSON. See Python_1000Q.json, Java_1000Q.json, and Javascript_1000Q.json in the provided repository for details. [7]

Section 3.2 – Gathering ChatGPT Output

The next phase of data collection involved taking the questions from the Stack Overflow posts, feeding them to the ChatGPT API, and inserting the responses into the JSON dataset for later analysis.

The main method used in gathering these ChatGPT responses is the OpenAI API. A script was developed to iterate through the set of Stack Overflow questions and feed the API for each of these questions. The textual result was saved in our data structure verbatim along with the date on which it was generated.

Section 3.3 – Application of measures; Readability, Cosine Similarity, and Moss Similarity

For analysis, the primary foci were cosine similarity and readability. The key was to find if there were any similarities and trends between both the Stack Overflow and ChatGPT generated answers.

Cosine similarity is a method of measuring the similarity of two vectors of attributes and returns a value between –1 and 1. A result of –1 would identify opposite vectors whereas a value of 1 would identify proportional vectors. Below is the formula used with cosine similarity, with attributes represented as A and B and theta as the similarity score.

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Formula 3-1: The formula for cosine similarity

As can be seen, the cosine similarity function is simply the dot product of the two vectors divided by each's magnitude, equal to cosine of theta. This data was gathered and displayed in bar graphs as seen in the results section.

Regarding this study, for every question pulled from Stack Overflow, one ChatGPT response was generated. For every ChatGPT response to a given question a cosine score was generated between the ChatGPT response and each member of the set of Stack Overflow answers associated with the question.

For readability, the module 'Textstat' in Python offered numerous methods in calculating readability, along with providing a few other statistics such as the quantity of complex words, syllables in the texts, and other general statistics on the text such as word count. Below are the formulas for each method provided in the results section.

$$206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right)$$

Formula 3-2: The formula for Flesch Reading Ease

Score	Difficulty
90-100	Very Easy
80-89	Easy
70-79	Fairly Easy
60-69	Standard
50-59	Fairly Difficult
30-49	Difficult
0-29	Very Confusing

Table 3-1: This table serves as a key for understanding the scores for the Flesch Read Ease method.

$$0.1579\left(\frac{difficult\ words}{words}\right) + 0.0496\left(\frac{words}{sentences}\right)$$

Formula 3-3: The formula for Dale-Chall Readability

Score	Understood by
4.9 or lower	average 4th-grade student or lower
5.0–5.9	average 5th or 6th-grade student
6.0–6.9	average 7th or 8th-grade student
7.0–7.9	average 9th or 10th-grade student
8.0–8.9	average 11th or 12th-grade student
9.0–9.9	average 13th to 15th-grade (college) student

Table 3-2: This table serves as a key for understanding the scores for the Dale-Chall method.

$$4.71\left(\frac{characters}{words}\right) + 0.5\left(\frac{words}{sentences}\right) - 21.43$$

Formula 3-4: The formula for Automated Readability Index (ARI)

A note for the ARI: the returned result represents the grade level. For instance, if the result reads 5, then it would be understood by a child in fifth grade, and 6.5 would represent education from sixth to seventh grade.

A measure of code similarity also needs to be considered. MOSS is the tool used here. MOSS is a type of copy-detection algorithm that determines the likeliness that one document has copied code from another. The algorithm employs mechanisms to ensure whitespace insensitivity, noise suppression, and position independence.

Instead of comparing every substring of a document to every substring of another, MOSS leverages document fingerprinting. A set of hashes is generated for each document, and comparisons are made between the fingerprints instead. This is done by first preprocessing the document to remove irrelevant features. A sequence of hashes of the k-grams of the preprocessed document are generated. Then a subset of hashes is selected for use as a document's fingerprint. Finally, pairs of documents with a high number of matching fingerprints are flagged.

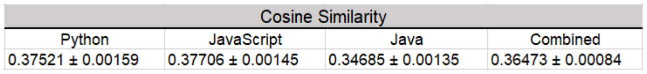
Documents that are flagged are paired and a line count is returned. This line count represents the number of lines that are similar. The individual documents within the pairing also have a percentage assigned. These percentages indicate the amount of the document that is like the other; the percentage of code that is shared from the point of view of the individual document in question.

4. DATA

This paper describes the data used in an analysis of ChatGPT's responses to 1100 questions on three programming topics - Python, JavaScript, and Java. The questions were chosen to cover a broad range of topics within each language, including syntax, data structures, algorithms, and libraries.

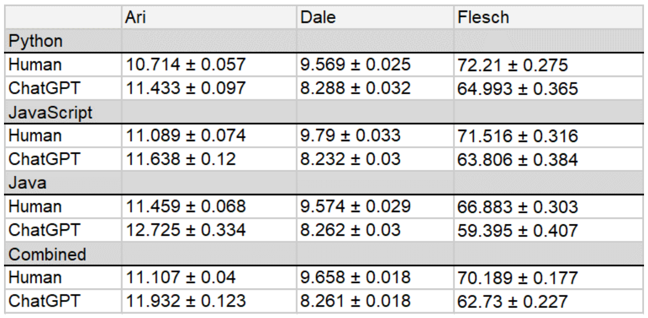
The questions selected represent the highest rated questions, thus are questions that are deemed commonly asked by the users. The dataset comprises the questions and the corresponding responses generated by ChatGPT. In this section, we will provide a detailed description of the dataset, including its size, structure, and the features used in the analysis.

We also calculated the mean and standard error of the mean for several measures (Fig 4-1, Fig 4-2), including cosine similarity, Flesch readability, Dale readability, and ARI readability. These measures allowed us to quantify the similarity between ChatGPT's responses and user responses on Stack Overflow, as well as the readability of these responses. The use of mean and standard error of the mean allowed us to estimate the average value of each measure and the variability around that average (Formula 4-1, Formula 4-2). This analysis enabled us to learn more about ChatGPT's responses and how it stacks up against Stack Overflow responses from users.



Cosine Similarity			
Python	JavaScript	Java	Combined
0.37521 ± 0.00159	0.37706 ± 0.00145	0.34685 ± 0.00135	0.36473 ± 0.00084

(Figure 4-1)



	Ari	Dale	Flesch
Python			
Human	10.714 ± 0.057	9.569 ± 0.025	72.21 ± 0.275
ChatGPT	11.433 ± 0.097	8.288 ± 0.032	64.993 ± 0.365
JavaScript			
Human	11.089 ± 0.074	9.79 ± 0.033	71.516 ± 0.316
ChatGPT	11.638 ± 0.12	8.232 ± 0.03	63.806 ± 0.384
Java			
Human	11.459 ± 0.068	9.574 ± 0.029	66.883 ± 0.303
ChatGPT	12.725 ± 0.334	8.262 ± 0.03	59.395 ± 0.407
Combined			
Human	11.107 ± 0.04	9.658 ± 0.018	70.189 ± 0.177
ChatGPT	11.932 ± 0.123	8.261 ± 0.018	62.73 ± 0.227

(Figure 4-2)

$$\text{Standard Deviation } \sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{N}}$$

(Formula 4-1)

$$\text{Standard Error of the Mean: } \frac{\sigma}{\sqrt{N}}$$

(Formula 4-2)

Figure 4-1 and 4-2: Preliminary measures of cosine similarity and readability scores.

Section 4.1 – Data Collection

For our research, we collected datasets of 1100 programming-related questions across three popular programming languages: Python, JavaScript, and Java. We obtained human-generated responses to each of these questions through Stack Overflow. In addition, we used the

OpenAI API to generate responses to each question using ChatGPT, a state-of-the-art language model. The resulting datasets allowed us to compare the similarity and readability of ChatGPT's responses to Stack Overflow responses for each programming language. The use of both human-generated responses from Stack Overflow and ChatGPT-generated responses provides a comprehensive view of the performance of ChatGPT as a tool for programming-related tasks.

Section 4.2- Dataset Size

The dataset is comprised of three JSON files, one for each programming language. Each language has 1100 entries of data, for a total of 3,300 ChatGPT answers to questions from Stack Overflow. Attached to this combination is a set of Stack Overflow answers for each question.

Section 4.3- Dataset Structure:

The schema of the JSON files are as follows:

- The outer set of keys are an identifier representing the question identifiers as defined in the Stack Exchange API. Each of these objects contains four keys: *stack question*, *stack answers*, *ChatGPT answers*, and *scoring*. These keys point to nested objects contain as follows:
- *stack question* contains a creation date, title, score, and body as defined in the Stack Exchange API.
- *stack answers* contain a set of keys which are answer identifiers as defined in the Stack Exchange API. Each of these keys contain the attributes *is_accepted*, *score*, *creation_date*, and *last_edit_date*.
- *ChatGPT answers* contain a tuple with two elements. The first element is text returned from a prompt of the ChatGPT API. The second element is the date on which it was generated.
- *scoring* contains the following keys: *cosine*, *moss*, *flesh*, *dale*, and *ari*, tracking relevant measurability scoring for the relevant identifiers involved.
-

5. TOOLS

This section describes the tools used in the research paper to analyze the data collected on ChatGPT's responses to programming-related questions. The analysis involved several tasks, including data cleaning, feature extraction, and visualization. To accomplish these tasks, we used a combination of Python programming language and various libraries, including the OpenAI API, JSON, and Matplotlib.

Section 5.1- Python

Python is a popular high-level programming language widely used for data analysis and scientific computing. It is known for its simplicity, readability, and powerful built-in libraries. We chose Python for this research paper because of its ease of use and availability of libraries specifically designed for data analysis.

Section 5.2- OpenAI API

The OpenAI API is a language model API that allows developers to access state-of-the-art natural language processing (NLP) models. In this research paper, we used the OpenAI API to generate responses to the programming-related questions posed to ChatGPT. The API enabled us to train ChatGPT on programming-related texts and generate responses that were specific to each programming language.

Section 5.3 - MOSS API

The MOSS API is a web service provided by Stanford which is a tool used to determine code similarity. It's often used for plagiarism detection. This tool is discussed in section 3.3. Mosspy is the interface used to access the service as an API and automate the querying of the service.

Section 5.4- JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format widely used for data exchange between different programming languages. We used JSON to store and exchange the data collected on ChatGPT's responses to programming-related questions. JSON's simplicity and flexibility made it easy to store and manipulate the data, facilitating the analysis process.

Section 5.5- Matplotlib

Matplotlib is a popular data visualization library in Python used to create a wide range of graphs and charts. In this research paper, we used Matplotlib to visualize the data collected on ChatGPT's responses to programming-related questions. Matplotlib's flexibility and ease of use enabled us to create informative and visually appealing graphs and charts, making it easier to present our findings.

6. RESULTS

Section 6.1- Cosine Similarity

For cosine similarity, a text extraction and code extraction tool were used to grab only the text-based segments within each answer. Thus, for these scores the code was disregarded. The cosine similarity was tested only on the extracted text. The same was done to the ChatGPT answers to maintain consistency when comparing the text extractions.

It should be noted that the code extraction only included lines of code which exist outside of a textual paragraph. Code snippets were maintained within a paragraph as they contribute to the overall semantics of the text involved. Consider the following:

"The meaning of a <code> yield </code> statement is…"

The yield keyword is maintained in this textual analysis. The code tags are also maintained, as this gives extra insight to the semantics of the word they surround. This also shores up the case in which code is included in-paragraph which extends beyond keywords.

The cosine similarity scores between Stack Overflow and ChatGPT answers per question were compared to evaluate the similarity of a higher and lower rated answer. The top five answers per question were evaluated among the three languages: Java (Figure 6-1), JavaScript (Figure 6-2), and Python (Figure 6-3).

It is worth noting that the results for the top five answers, denoted in figures 6-4, 6-5, and 6-6 were conducted on a sorting algorithm based on the cosine similarity score. Thus, on a scale of highest to lowest cosine scores, the 600th question's highest rated scored at around 0.4 similarity. Relatively consistent across all languages, the top answer per question had roughly a higher score than the fifth question, denoting top answers are most like ChatGPT answers.

The top Stack Overflow answers were compared to evaluate the relative difference between scores of Java, Python, and JavaScript, as seen in figure 6-7. The scores were sorted from highest to lowest scores to indicate the overall denotation of the scores. Although each language is relatively similar in scores, java tends to have higher scores compared to JavaScript.

While looking at figures 6-4 through 6-7, it's worth noting the distribution of ratings given on Stack Overflow. The range of quartiles of the ratings in the aggregated set are: Q1: 3, Q2: 10, Q3: 43, and Q4: 34634. The quartiles on the individual datasets are as follows:
- Python: Q1: 4, Q2: 15, Q3: 70, and Q4: 17604, Min: -36
- Java: Q1: 2, Q2: 8, Q3: 34, Q4: 34634, Min: -57
- JavaScript: Q1: 2, Q2: 8, Q3: 36, Q4: 16150, Min: -74

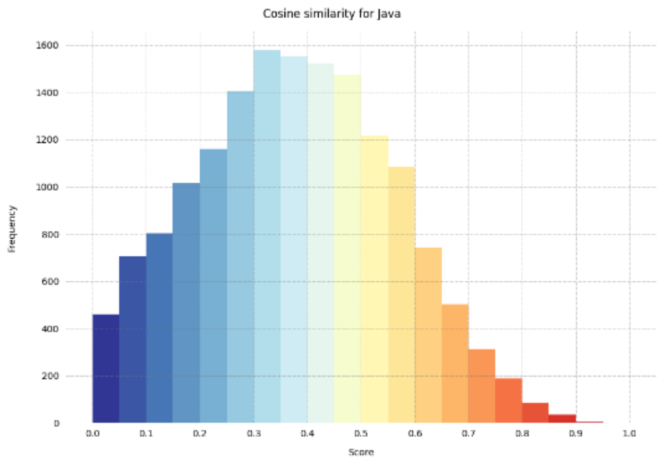


Figure 6-1: A histogram representing the cosine similarity scores comparing ChatGPT answers to Stack Overflow answers for the same prompt within the Java tag. The x axis represents the cosine similarity score, and the y axis represents the frequency of answers.

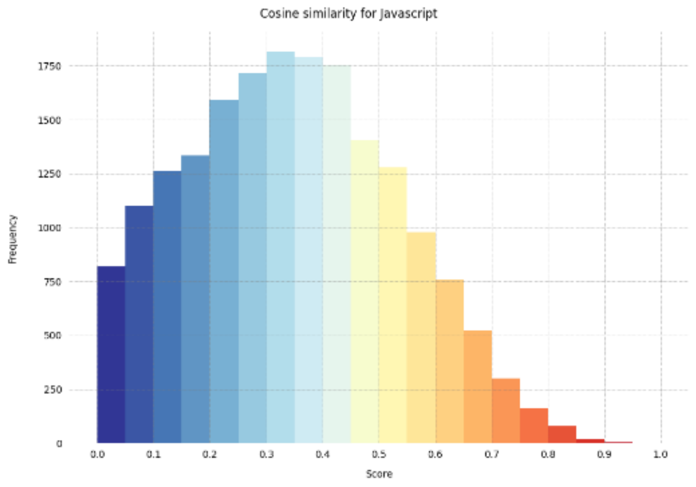


Figure 6-2: A histogram representing the cosine similarity scores when comparing ChatGPT answers to Stack Overflow answers for the same prompt within the JavaScript tag. The x axis represents the cosine similarity score, and the y axis represents the frequency of answers.

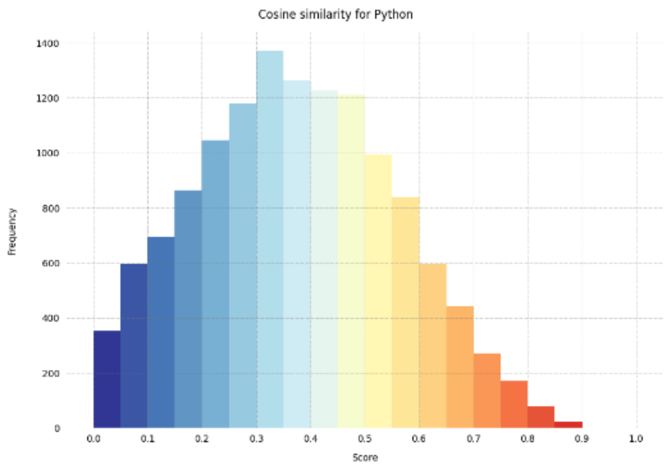


Figure 6-3: A histogram representing the cosine similarity scores when comparing ChatGPT answers to Stack Overflow answers for the same prompt within the Python tag. The x axis represents the cosine similarity score, and the y axis represents the frequency of answers.

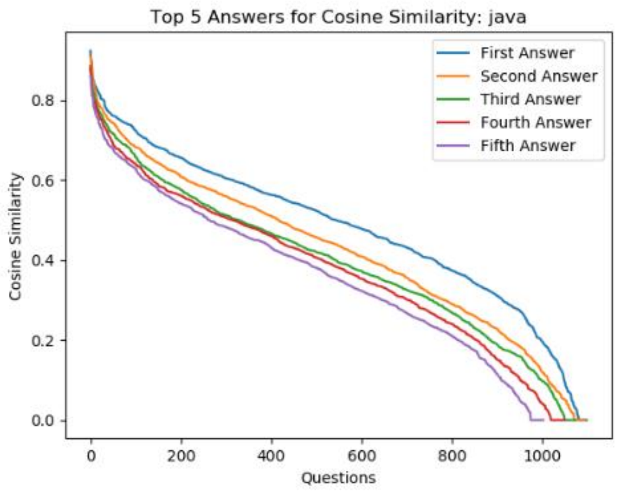


Figure 6-4: Cosine similarity for each question's top-five-scored answers when ordered from highest to lowest score for Java tagged posts in Stack Overflow. The x axis represents each question, and the y axis represents the cosine similarity score.

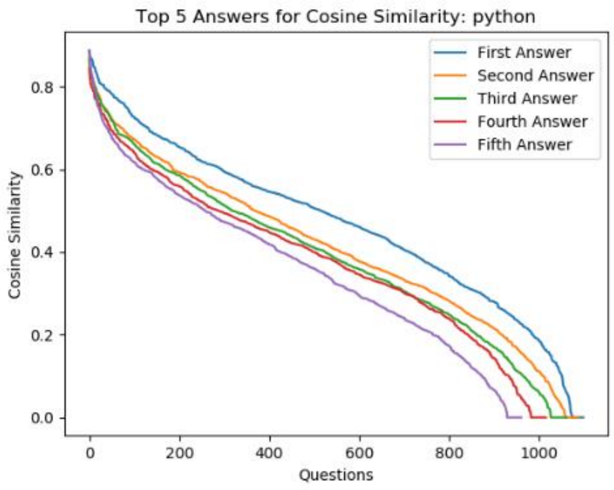


Figure 6-6 Cosine similarity for each question's top-five-scored answers when ordered from highest to lowest score for Python tagged posts in Stack Overflow. The x axis represents each question, and the y axis represents the cosine similarity score.

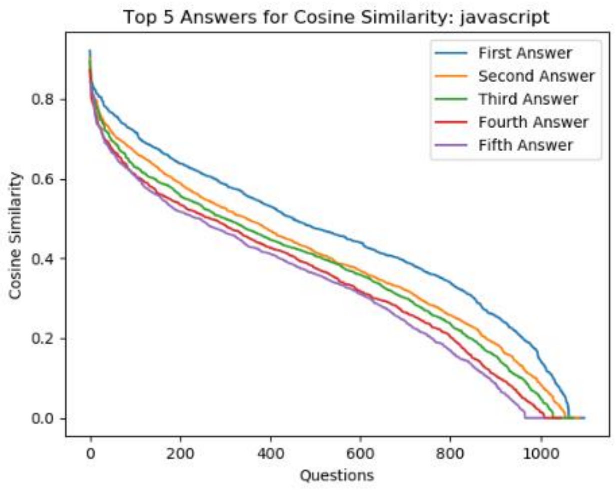


Figure 6-5 Cosine similarity for each question's top-five-scored answers when ordered from highest to lowest score for Javascript tagged posts in Stack Overflow. The x axis represents each question, and the y axis represents the cosine similarity score.

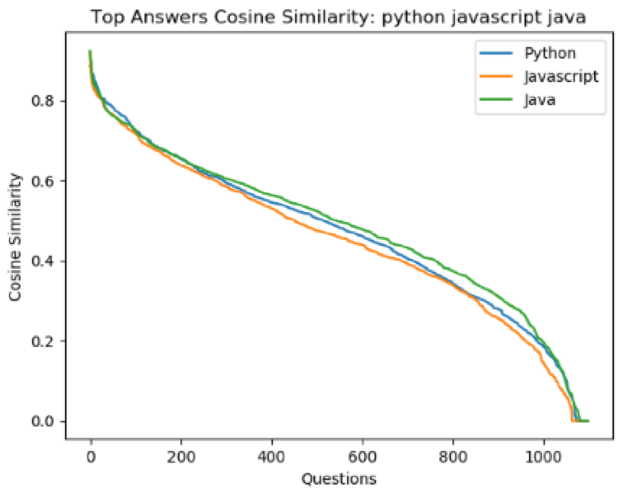


Figure 6-7: Cosine similarity for each question's top-scored answers when ordered from highest to lowest score for Java, Javascript, and Python tagged posts in Stack Overflow. The x axis represents each question, and the y axis represents the cosine similarity score.

Section 6.2- Dale-Chall Formula

The Dale-Chall Formula under the Textstat library, is used to determine readability scores for ChatGPT vs Stack Overflow. The Dale Chall incorporates a 3000 word look up table and returns a relative grade level for the text document. Textual preprocessing, as described in section 6.1, was applied when evaluating these scores. For languages such as Java, Dale-Chall scores for Stack Overflow and ChatGPT were most frequent between scores 8.0-10.0 as seen in Figure 6-8. Languages such as Python and JavaScript included similar results averaging around 8.0-10.0. The frequency of Stack Overflow scores in total are larger than those of ChatGPT due to the larger number of stack overflow answers used per question.

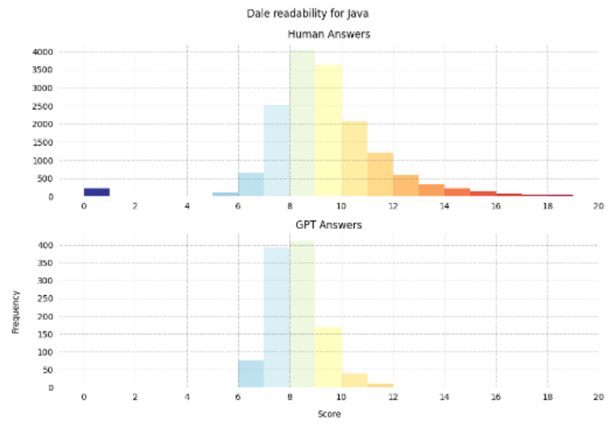


Figure 6-8: A comparison of Dale-Chall readability between Stack Overflow user answers and ChatGPT answers, with the Dale-Chall score on the x axis and frequency on the y axis.

Section 6.3- Flesch Reading Ease

The Flesch reading ease test under the Textstat library is an aid in determining relative readability. Higher scores indicate easier material, and lower scores determine material directed toward professionals. Like cosine similarity, the results were conducted on a sorting algorithm based on the Flesch reading score. Textual preprocessing, as described in section 6.1, was applied when evaluating these scores. The Flesch reading ease test demonstrated peak frequency of scores around 60-70 for Java in both Stack Overflow and ChatGPT answers as shown in figure 6-9. Languages such as Python and JavaScript peaked closer to 70, with denotes that Python and JavaScript have answers that are easier to read opposed to Java, according to Flesch reading ease. The frequency of Stack Overflow scores in total are larger than those of ChatGPT due to the larger number of stack overflow answers used per question.

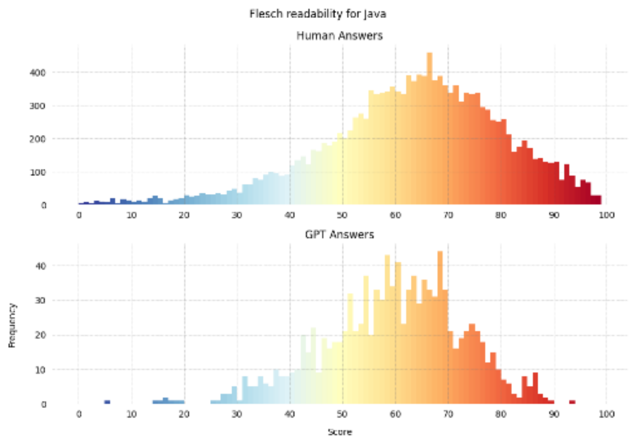


Figure 6-9: A comparison of Flesch readability between Stack Overflow user answers and ChatGPT answers, with the scores on the x axis and frequency on the y axis.

Section 6.4- Automated Readability Index

The Automated Readability Index, ARI, under the Textstat library is an aid in determining relative readability. The ARI returns a grade level which is needed to understand the text document. Textual preprocessing, as described in section 6.1, was applied when evaluating these scores. Stack Overflow responses for the language, Java, had a peak frequency count around 10, and ChatGPT peaked slightly higher around a score of 13, demonstrated in figure 6-10. Languages such as Python and JavaScript peaked in frequency count around a score of 10-11 for both ChatGPT and Stack Overflow answers. Python and JavaScript had a greater number of scores between 15-20 for ChatGPT answers than the ChatGPT answers had for java. The frequency of Stack Overflow scores in total are larger than those of ChatGPT due to the larger number of stack overflow answers used per question.

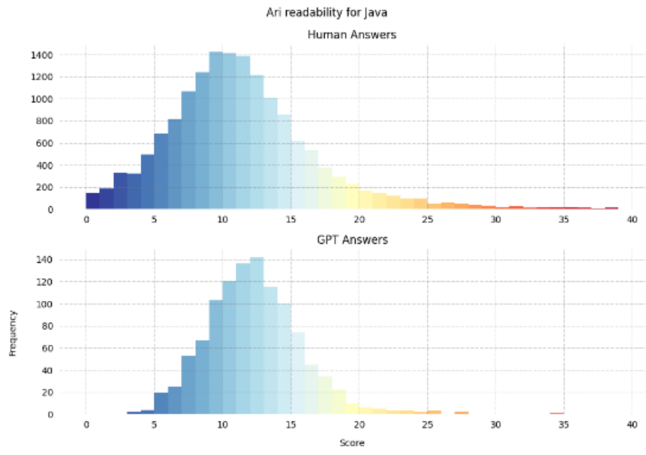


Figure 6-10: A comparison of ARI readability between Stack Overflow user answers and ChatGPT answers s, with the ARI score on the x axis and frequency on the y axis.

Section 6.5- MOSS scoring

To complement section 6.1, MOSS scoring is an attempt at evaluating the semantics of the code involved in an answer. The code which wasn't embedded in a paragraph was evaluated for MOSS scoring. Consider the following:

```
"Here is an example of an <code>if</code> statement:
 <code>
   string = 'hello'
   if string == 'hello':
         print('world!')
 </code>"
```

In the above example, only the content *between* the second set of code tags is isolated and evaluated. This preprocessing applies to both the ChatGPT generated solution and all Stack Overflow solutions.

Results here are sparse. Whilst looking at the Python dataset. Of the 1100 ChatGPT solutions, 1064 of them contained code exclusive from paragraph content. Only 88 of these solutions were flagged for code similarity. Some of these solutions had multiple flags within their respective set of Stack Overflow solutions, leading to a total of 131 flags for code similarity.

To provide further analysis, the quartiles of all the ratings associated with Stack Overflow answers were generated. The range of these quartiles are [-36,4], [5,15], [16, 70], [70, 17604]. Of the 134 moss scores, 13 were associated with Stack Overflow answers with ratings in the first quartile. 28 were associated with answers in the second quartile. 40 with answers in the third quartile, and 53 with answers in the fourth quartile.

Data was not gathered for Moss scores with respect to Java and JavaScript tags. This is discussed further in limitations.

7. ANALYSIS

In this section, we present the results of our analysis of ChatGPT's responses to programming-related questions. We focus on two primary measures of performance: readability and cosine similarity. We compare the readability scores of ChatGPT's responses to those of human-generated responses from Stack Overflow and examine the cosine similarity between the two sets of responses. Finally, we draw higher-level inferences from these results. Based on our analysis, we draw the following higher-level inferences:

ChatGPT can generate responses to programming-related questions that are similar in readability to Stack Overflow responses. However, there is room for improvement in ChatGPT's response generation to make it easier to read when compared to what is typical of Stack Overflow.

The cosine similarity measures show that ChatGPT's responses are very similar to human-generated responses that have a higher rating on Stack Overflow. This indicates that ChatGPT can accurately predict programming concepts and can generate responses that are those that are favored on Stack Overflow.

The sparse results given for MOSS scoring don't lead to meaningful analysis. It is a question whether this is in part to the (lack of) sensitivity the algorithm provides or if more data needs to be gathered. The former is more likely – when counting the amount of MOSS similarity flags determining whether a solution to Stack Overflow question is similar to another solution to the same question, the flag count is pushed from 134 to 783. To give context, the total amount of solutions in the Python dataset is 13,958.

It is worth noting that of the MOSS flags related to a ChatGPT solution, more than two-thirds of them have a rating that is greater than 16; they lie in either the third or fourth quartile of the ratings distribution.

8. LIMITATIONS

There were a few limitations in gathering the data for this study, much of which had to do with time, number of resources, and the capabilities of the resources for analyzing the data. One such limitation was Open AI's API, which was slow to generate answers for the Stack Overflow questions. Also, this resource required payment after a free trial usage, so we were only able to gather data for a few thousand questions. That said, the amount of data that was gathered did provide good insight to how ChatGPT answers compare to Stack Overflower's user generated answers. Based on the trends shown above, more data would help solidify those claims.

Similarly, the API we used for determining MOSS similarity was not timely in generating results. The web service offers no documentation on call limits. The thresholds of throttling the calls per API key and per IP address are unclear. With a script that cycles through a set of five API keys in alternating order, the number of calls per day ranged from 100 to 250. Due to the time constraints of this project and the need to sequentially produce ChatGPT answers, we did not have time to measure MOSS scores for the Java and JavaScript datasets.

With more time and resources, a larger scale of this study could be carried out, providing more conclusive data for analysis.

When considering the behaviors of Stack Overflow users, it should be noted that users who post less frequently will often respond to more complex problems with an accepted response, whereas more common users make up a significant number of accepted responses. This variety in behavior could produce varying responses to post questions; however, ChatGPT can be regarded as a single entity on its own. This single entity has a style and approach regardless of the question. It is currently unclear the extent to which this affects interpreted accuracy to the semantics of a given answer.

9. CONCLUSIONS

ChatGPT is a relatively new deployment. There is still a lot of uncertainty regarding its ability to produce readable and similar results. This study aims at looking at the gap between solutions provided by it and those expected through Stack Overflow.

We analyzed ChatGPT's responses to programming-related questions for three programming languages: Python, JavaScript, and Java. Our analysis focused on two primary measures of performance: readability and cosine similarity. We compared ChatGPT's responses to these Stack Overflow responses from users and drew higher-level inferences from the results.

Our analysis showed that ChatGPT's responses had similar readability scores to the user responses which had a higher rating on the platform, indicating that ChatGPT can generate responses that are as easy to comprehend. However, there is room for improvement in ChatGPT's response generation to make this easier. The cosine similarity measure showed that ChatGPT's responses are highly alike to these user responses which had a higher rating on the platform, indicating that ChatGPT can predict programming concepts and can generate responses that are similar and complete.

These findings have important implications for the field of programming and natural language processing. ChatGPT's ability to generate similar and complete responses to programming-related questions has significant potential for use in education, technical support, and other applications where programming knowledge is necessary. Additionally, the findings of our study highlight the need for ongoing research to improve the readability of ChatGPT's responses, enabling it to generate responses that are both similar to Stack Overflow and easy to read.

In conclusion, our analysis of ChatGPT's responses to programming-related questions provides valuable insights into the potential of ChatGPT in the field of programming and natural language processing. We hope that our findings will inspire further research in this area and contribute to the development of more sophisticated and effective natural language processing systems.

10. CONTRIBUTIONS

Alan McKay

Developed the primary script used to gather information from Stack Overflow. This includes the development of the schema used to contain our data while also declaring which elements of an API call are important.

Developed the primary script used to interface with the ChatGPT API including how this script interfaces with the data structures involved. Gathered the ChatGPT solutions for both the Python and JavaScript dataset.

Developed the script that interfaces with the MOSS API. This includes how the script interfaces with the data structures involved. Sole contributor to any sections pertaining to MOSS within this research paper.

Developed the helper functions used to preprocess the data for score evaluation.

Gathered the sources used in the related works section. This includes getting a good feel for what the papers are about and how they are related to this research.

Guided the team’s progress and decision making in terms of how to handle the data involved.

Hayden Westphal

Hayden played a key role in the data collection phase of this research project, where Hayden helped to collect a dataset of 1,000 programming-related questions for one of the three topics. In addition, Hayden also contributed to the analysis of the data by developing a Python script that utilized matplotlib to create histograms for the various measures that we calculated, including cosine similarity, Flesch readability, Dale-Chall readability, and ARI readability. These histograms were essential in providing a visual representation of the data and helped to support the conclusions that we drew from the analysis.

Jordyn Iannuzzelli

Throughout the project, Jordyn has contributed to the overall planning behind the entirety of the project as well as the deliverable submissions. A focus Jordyn has worked on in prior deliverables was finding libraries that can help generate scores for cosine similarity and readability metrics. Since the last deliverable, Jordyn has worked to gather all the cosine similarity scores, Dale-Chall scores, Flesch Reading scores, and Automated Readability Index scores for the generated questions and answers. Jordyn has contributed to producing resulting graphs to display some of the logistics behind the scores across the three languages.

Nathan Hine

Overall, Nathan helped generate ideas for how to tackle several roadblocks throughout the project, from gathering data from ChatGPT to analyzing the data. With ChatGPT, he helped to discover the OpenAI API and work the functions within to generate answers for the proposed questions from Stack Overflow, which was fine tuned into a working data collection system with help from the rest of the group. Nathan also tackled the methods and limitation sections of the paper, researching the formulas and various methods used for data analyzation (not including Moss similarity, that was Alan). He helped with other smaller parts of the paper, such as consistency in writing and formatting. Since the last deliverable, Nathan has helped with composing the paper, ensuring consistency in formatting throughout, aiding in generating parameters for data presentation, and aiding in small tasks with content for the paper.

11. REFERENCES

[1] OpenAI. 2021. ChatGPT. OpenAI Blog. (2021). https://openai.com/blog/chatting-with-ai/

[2] Stack Exchange. 2019. Stack Exchange API Documentation. https://api.stackexchange.com/docs

[3] BulkGPT: Bulk Prompts for GPT-3 (2023). Retrieved from https://chrome.google.com/webstore/detail/bulkgpt-bulk-prompts-for/emamaaonhmidkfldhlmpfgmdgjgonhmg

[4] Schleimer, S., Wilkerson, D., &amp; Aiken, A. (2003, June). Winnowing: local algorithms for document fingerprinting. ACM Digital Library. Retrieved April 25, 2023, from https://dl-acm-org.proxy.lib.uiowa.edu/doi/abs/10.1145/872757.

[5] Wang, S., Chen, T.-H., & Hassan, A. E. (2018). Understanding the factors for fast answers in technical Q&A websites: An empirical study of four stack exchange websites. Empirical Software Engineering: An International Journal, 23(3), 1552-1593.

[6] Aiken, A. (n.d.). Moss (Measure of Software Similarity). Retrieved March 7, 2023, from https://theory.stanford.edu/~aiken/moss/

[7] McKay, A., Westphal, H., Iannuzzelli, J., &amp; Hine, N. (2023, March 1). web_mining_project. GitHub. Retrieved April 26, 2023, from https://github.com/alanmmckay/web_mining_project.git