2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC) | 979-8-3503-2697-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/COMPSAC57700.2023.00117

# Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data

Yunhe Feng†, Sreecharan Vanam†, Manasa Cherukupally†, Weijian Zheng‡, Meikang Qiu§ and Haihua Chen†
†University of North Texas, ‡Argonne National Laboratory, §Dakota State University
†yunhe.feng@unt.edu, †vanamsreecharan@my.unt.edu, †manasacherukupally@my.unt.edu
‡wzheng@anl.gov, §meikang.qiu@dsu.edu, †haihua.chen@unt.edu

***Abstract*—The recent advancements in Artificial Intelligence, particularly in large language models and generative models, are reshaping the field of software engineering by enabling innovative ways of performing various tasks, such as programming, debugging, and testing. However, few existing works have thoroughly explored the potential of AI in code generation and users' attitudes toward AI-assisted coding tools. This knowledge gap leaves it unclear how AI is transforming software engineering and programming education. This paper presents a scalable crowdsourcing data-driven framework to investigate the code generation performance of generative large language models from diverse perspectives across multiple social media platforms. Specifically, we utilize ChatGPT, a popular generative large language model, as a representative example to reveal its insights and patterns in code generation. First, we propose a hybrid keyword word expansion method that integrates words suggested by topic modeling and expert knowledge to filter relevant social posts of interest on Twitter and Reddit. Then we collect 316K tweets and 3.2K Reddit posts about ChatGPT's code generation, spanning from Dec. 1, 2022 to January 31, 2023. Our data analytics show that ChatGPT has been used in more than 10 programming languages, with Python and JavaScript being the two most popular, for a diverse range of tasks such as code debugging, interview preparation, and academic assignment solving. Surprisingly, our analysis shows that fear is the dominant emotion associated with ChatGPT's code generation, overshadowing emotions of happiness, anger, surprise, and sadness. Furthermore, we construct a ChatGPT prompt and corresponding code dataset by analyzing the screenshots of ChatGPT code generation shared on social media. This dataset enables us to evaluate the quality of the generated code, and we have released this dataset to the public. We believe the insights gained from our work will provide valuable guidance for future research on AI-powered code generation.**

***Index Terms*—ChatGPT, Coding Generation, Software Engineering, Large Language Models (LLMs), Generative Models, Social Media**

## I. Introduction

Recently, the advancements in large language models (LLMs) and generative models have revolutionized many applications, including free text generation, question answering, and document summarization, enabling a wide range of real-world services such as AI robot lawyers [1] and AI music co-creation [2]. The field of coding, which involves writing tasks in certain programming languages, is also benefiting from the rapid development of generative LLMs. However, unlike traditional writing tasks, programming requires strict adherence to syntax and logic rules, making it more challenging for generative models to produce high-quality code.

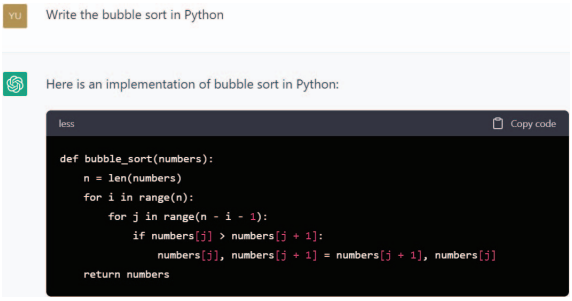



Fig. 1. ChatGPT writes the bubble sort algorithm in Python

Several studies have investigated the potential of LLMs in software development. For instance, Barke et al. [3] and Vaithilingam et al. [4] examined user perceptions of generative models in coding writing. However, many of these studies are based on case studies, with limited consideration of broader applications in software development. The emerging OpenAI's ChatGPT, a member of the GPT LLM family, demonstrates promising performance in code generation, attracting widespread attention from stakeholders in software engineering. As shown in Figure 1, ChatGPT can generate the bubble sort algorithm in Python with the prompt of "write the bubble sort in Python." Some studies have explored the use of ChatGPT for code generation tasks [5]–[7]. Nonetheless, these studies did not comprehensively evaluate the overall effectiveness of ChatGPT as a code generation and assistance tool on a large scale.

It is challenging to conduct a large-scale study on the performance of LLMs in code generation due to the following reasons. First, programming languages exhibit diverse syntax and are applicable to a wide range of tasks. For instance, SQL is primarily utilized in database operations, while JavaScript is commonly used in web programming. Second, code generation encompasses numerous programming tasks, including debugging, testing, and programming, for various stakeholders. Moreover, conducting user studies in the lab to investigate the code generation of LLMs can be costly and time-consuming. Therefore, conducting a comprehensive study on the performance of LLMs that covers numerous programming languages, tasks, and stakeholders poses significant challenges.

To address the aforementioned challenges, this paper pro-

979-8-3503-2697-0/23/$31.00 ©2023 IEEE
DOI 10.1109/COMPSAC57700.2023.00117

poses a scalable crowdsourcing data-driven framework that integrates multiple social media data sources to examine the code generation performance of ChatGPT. The proposed framework comprises three key components, namely keyword expansion, data collection, and data analytics. Specifically, we utilize topic modeling and expert knowledge to identify all keywords that are relevant to programming in the context of ChatGPT, thus expanding the seed keyword of *ChatGPT*. Using these expanded keywords, we retrieved 316K tweets and 3.2K Reddit posts related to ChatGPT's code generation from December 1, 2022, to January 31, 2023.

Furthermore, we conduct a comprehensive analysis using multimodal data (text and images) to answer the following research questions:

1) What are the most popular programming languages in ChatGPT usage?
2) What programming scenarios, tasks, and purposes are people using ChatGPT for?
3) What is the temporal distribution of the discussion on ChatGPT code generation?
4) How do stakeholders perceive ChatGPT code generation?
5) What are the prompts to generate code?
6) What is the quality of the code generated by ChatGPT?
7) Does the generated code present any ethical issues?

To the best of our knowledge, this work is the first large-scale, systematic study on emerging generative models for code writing and testing using crowdsourced social data. We summarize our contributions as follows:

- We have proposed a scalable crowdsourcing and social data-driven framework for investigating the code generation capabilities of ChatGPT.
- We have presented a novel hybrid keyword expansion method that incorporates words recommended by topic modeling and experts to ensure that most of the related social media posts are matched during data collection.
- Our study considers multiple social media platforms (Twitter and Reddit) and multimodal data (text and image) to mitigate potential biases caused by a single data source or data type.
- We have provided data analytics from multiple perspectives, including topic inference, sentiment analysis, and data quality measurement.
- We have built a real-world programming dataset containing the ChatGPT prompt and the associated generated Python code. This dataset is publicly available at https://shorturl.at/oEMN2.

## II. Related Work

**Automatic Code Generation.** Many machine learning and deep learning models [8], [9] have been explored for automatic programming. For example, Raychev et al. [10] proposed a code completion technique using statistical language models to discover highly rated sentences and recommend code completion suggestions. Sun et al. [11] introduced a novel tree-based neural architecture that incorporates grammar rules and abstract syntax tree structures into the network, and it was reported to achieve the best accuracy among all neural network-based code generation methods. Ciniselli et al. [12] conducted a detailed empirical study on BERT models for code completion and evaluated the percentage of perfect predictions that match developer-written code snippets.

As ChatGPT has gained more attention recently, some researchers have studied its use for code generation [5]–[7]. For example, Aljanabi et al. [5] listed automatic code generation as one of the open possibilities for ChatGPT. Avila et al. [6] elaborated on the programming potential of ChatGPT for implementing online behavioral tasks, including concurrent reinforcement schedules, using HTML, CSS, and JavaScript code. They created files with the extensions .html, .css, and .js, encompassing fundamental page structures like headings, style element integration, and dynamic components.

**Automatic Bug Fixing.** Unidentified and unsolved bugs in complex coding are always threatening the correctness and resilience of software systems. To automatically detect and fix code bugs and errors, the concept of Automated Program Repair (APR) has been proposed. Recent advancements in deep learning have facilitated the integration of APR into many large language models (LLMs). LLM-based tools such as Codex [13], CodeBERT [14], and Conversational APR [15] have been proposed for bug fixing.

A recent study [16] conducted a comparative evaluation of ChatGPT's efficiency in bug fixing with other baseline tools, such as Codex [13]. About 40 of QuixBugs benchmark problems containing erroneous code were given to ChatGPT to provide solutions. The experiment results showed that ChatGPT's performance was similar to other APR tools like Codex. However, when given more context information about the problem through its dialogue box, ChatGPT's performance improved, delivering a success rate of 77.5%.

**Interactions and Limitations.** As programming generation and assistant tools, such as CodeBERT [14] and IntelliCode Compose [17], become more widely used, there has been an increased focus on investigating the usability and interactions between users and code generation tools [3], [4], [18], [19]. For example, Barke et al. [3] identified two interaction modes between programmers and code generation tools: acceleration mode and exploration mode, by observing how 20 programmers solved various tasks using the code generation tool Copilot. Vaithilingam et al. [4] performed a study on 24 participants consisting of different groups of people with minimal and moderate experience in using Copilot and IntelliSense. By quantitative and qualitative analysis, they observed that participants who used Copilot failed to complete tasks more.

Although advanced automatic code generation tools work fine with simple code logic, it can be challenging to handle large software engineering projects [20]. For instance, the development of a web browser involves a deep understanding of human needs that are challenging to encapsulate within the confines of simple, machine-readable specifications, which AI typically employs to produce code [20]. In addition, ethical concerns about code generation models have begun to surface.

For example, Chatterjee and Dethlefs [21] found evidence of gender and racial bias in the code generated by ChatGPT, raising serious questions about the responsibility and fairness of such models.

Different from most existing works, we collect and employ large-scale datasets collected from multiple social media platforms to evaluate the coding performance of a general-purpose conversation tool, i.e., ChatGPT. In addition to investigating user responses towards ChatGPT's coding capabilities, our study also examines the programming tasks facilitated by ChatGPT and the ethical concerns of ChatGPT.

## III. Method

This section presents the proposed scalable crowdsourcing data-driven framework by introducing how to collect data of interest, how to analyze data, and how to interpret findings.

### A. Overview of the Proposed Framework

Figure 2 presents the overview of the proposed framework. It consists of three primary components: Keyword Expansion and Selection, Data Collection, and Data Analytics and Pattern Recognition. Contrasting with the traditional user study oriented research, crowdsourcing frameworks are more flexible and scalable, facilitating the examination of a large population over an extended time frame [22]. We will explore each component thoroughly, assessing the efficacy of LLMs in the realm of code generation.

### B. Keyword Selection for Software Development

To ensure the quality of the collected data, we employ a hybrid approach that combines data-driven keyword expansion and expert-based keyword selection. This approach ensures that the data is comprehensive and precise, eliminating the risk of bias or incompleteness in the selection of query keywords.

As ChatGPT is one of the most popular LLMs that supports code generation, we use *ChatGPT* as the seed keyword to sample Twitter streams, harvesting tweets that mention this term. We then perform topic modeling to determine whether a coding-related topic is present. If a coding-related topic is observed, we add the words belonging to this topic to the expanded keyword set. If a coding-related topic is not observed, we conduct a co-occurrence word analysis and calculate the semantic similarity with the word *coding* to expand the candidate keywords.

However, the data-driven keyword expansion method may result in false positives, i.e., keyword candidates irrelevant to AI-based code generation may also be included. Therefore, we manually examine all recommended keyword candidates to ensure the quality of the collected data. We first filter out irrelevant keywords and propose multiple combinations of keywords to control the precision of data collection. For example, instead of collecting all postings containing *ChatGPT*, collecting postings containing both *ChatGPT* and *coding* makes the retrieved data more accurate and representative.

Specifically, we leverage Twitter Streaming APIs to sample tweet streams containing the keyword *ChatGPT* for over 55 hours. In total, we collect 158,452 tweets, including original tweets, retweets, and replies. After removing duplicate tweets, we had 63,716 unique tweets. We then apply the latent Dirichlet allocation (LDA) [23] model to infer topics based on these unique tweets, with the hope of discovering programming-related topics. We evaluate the number of topics ranging from 1 to 30 and find that the convergence score achieves a relatively high and stable value with the number of topics set as 22. For more details, please see Figure 12. After examining the 22 topics, we identify one of them as "Programming," consisting of the following words: *ask, stack, knew, write, error, diffus, run, python, stabl, scientist, email, straight, shock, gener, comput, command, use, code, notic, brain, bug, statement, think, dead, question, admit, happen, result, and overflow*.

Combining the words in the topic of *Programming*, we come up with the following keyword list – *algorithm, algorithms, bug, bugs, c#, c++, code, coding, command, commands, compiler, computing, debug, debugging, error, interpreter, java, javascript, libraries, php, program, programming, python, r, Ruby, shell, software, sql, stack overflow, swift, test, testing, typescript* – to crawl ChatGPT related code generation posts.

### C. Data Collection

Based on the above carefully curated keywords, we leverage two social media platforms, i.e., Twitter and Reddit to collect data for further analytics.

*1) Twitter Data:* Instead of relying on Twitter Streaming APIs, we opt to use the Twitter Historical Data Search APIs to create our Twitter dataset for the following reasons: 1) The streaming data is time-sensitive, making it impossible to retrieve older data from the debut of ChatGPT if the streaming data collection was not be launched at that time; 2) Examining only the latest data (e.g., after Feb 1, 2023) could introduce bias, as we cannot determine when ChatGPT's code generation performance was most widely discussed on social media. On the other hand, the historical tweets span the entire evolution of ChatGPT and provide a sample of user comments since its release, enhancing the representativeness and completeness of the crowdsourced opinions and feedback.

Twitter provides two APIs that allow searching for historical data: the 30-Day Search API[1] enables access to data from the previous 30 days, while the Full-archive Search API[2] permits access to tweets from as far back as 2006, the year the first tweet was made. Given that ChatGPT was first introduced on November 30, 2022, we choose to use the Full-archive Search API to extract data. We specifically utilized Twitter's Academic Research API, known for its capability of executing full-archive tweet searches, to gather data related to ChatGPT from November 30, 2022, to February 1, 2023. Our search was meticulously designed only to include English tweets and exclude retweets, as indicated by the parameter "-is:retweet lang:en." In addition to the text of the tweets, we also collect

[1]https://tinyurl.com/2s4xt8r7
[2]https://tinyurl.com/ehbsjx6v

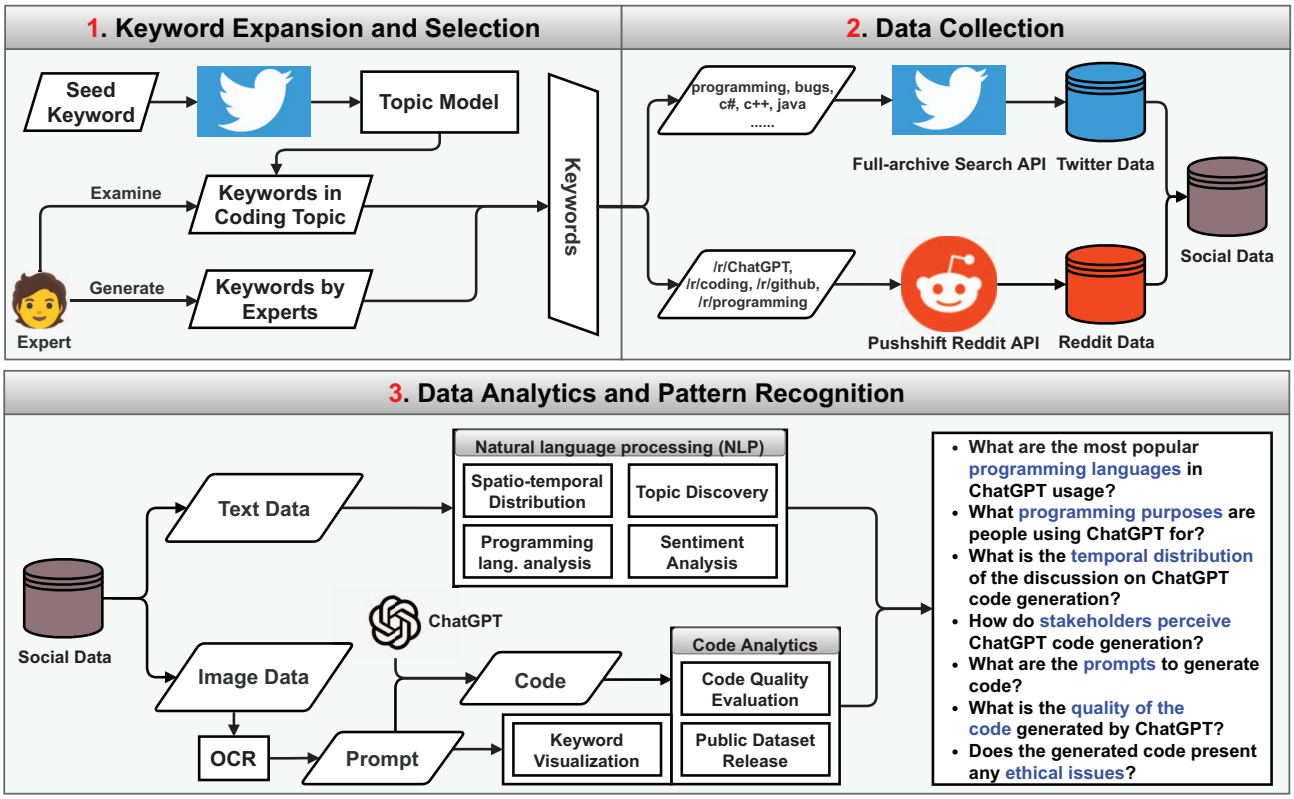



Fig. 2. Overview of the proposed crowdsourcing framework to investigate the programming capabilities of ChatGPT

related media information from Twitter, including images, to enhance our analysis. For this study, we compiled a total of 316K tweets posted between December 1, 2022, and January 31, 2023.

*2) Reddit Data:* Unlike Twitter, where the structure is based on users following one another, Reddit is structured around communities where posts on similar topics are grouped together. These communities are referred to as "subreddits" on Reddit. For instance, the subreddit /r/aww is a community where users share cute and cuddly pictures. The initial posts on Reddit are known as "submissions," and the responses to these posts are called "comments."

To assess the code generation ability of ChatGPT within the Reddit community, our attention is concentrated on four notable subreddits: /r/ChatGPT, /r/coding, /r/github, and /r/programming. We gather posts from these subreddits using the Search Reddit Submissions Endpoint (/reddit/search/submission) available through the Pushshift Reddit API [24]. Just like with Twitter data, we also collect multimedia data such as images embedded in Reddit posts. For the sake of this research, we have compiled 3.2K Reddit posts made between December 1, 2022, and January 31, 2023, to examine ChatGPT's code generation proficiency.

### D. Data Analytics and Pattern Recognition

We deploy natural language processing and image understanding techniques to uncover insights and identify patterns.

*1) Text Based Topic Discovery:* To attain a comprehensive understanding of ChatGPT's deployment in code generation across social media platforms, we resort to latent Dirichlet allocation (LDA) [23], a widely utilized technique for topic modeling. This approach reveals underlying topics hidden within the collected tweets and Reddit posts. Each tweet or post is regarded as an individual document, while the entire assembly forms the corpus. During text preprocessing, we implement standard procedures such as discarding stop words and frequently occurring terms like *ChatGPT*, in addition to tokenizing and lemmatizing words. We then carry out a term frequency-inverse document frequency (TF-IDF) analysis on the collated documents to construct a TF-IDF-based corpus. LDA models are subsequently employed to unearth latent topics within this corpus. To determine the optimal number of topics, we use the $C_v$ metric, consistent with prior research centered on large-scale social data analysis [25], [26]. This metric, which amalgamates normalized pointwise mutual information (NPMI) and cosine similarity [27], is acknowledged as one of the most efficacious coherence measures.

Given that Twitter allows users to utilize #hashtags to indicate related topics and enhance visibility through searches,

we also present the distribution of #hashtags in the collected tweets. However, as #hashtags are rarely used on Reddit, we do not perform this analysis for Reddit submissions.

*2) Image Understanding:* Given that ChatGPT operates as a text generation model, it is expected that a majority of images associated with it, especially those pertaining to code generation, shared across social media platforms, will be text-rich. To augment the practicality of these images and streamline their processing for subsequent tasks, we suggest the deployment of an Optical Character Recognition (OCR) technique to transmute the assembled images into text. We evaluate multiple OCR methodologies, including the OpenCV-backed pytesseract[3] and the deep learning-based easyOCR[4], on our image dataset. Following a thorough evaluation of the OCR detection results, we choose easyOCR as the tool for the precise identification and extraction of text from the images.

*3) Code Reconstruction from Image:* To reconstruct the code generated by ChatGPT, it is crucial to identify the images that contain generated code. After examining the screenshots of coding snippets, we found that all ChatGPT-generated code snippets contained the keyword *Copy code* in the top-right corner of the coding block, as shown in Figure 1. Therefore, we select all images containing the *Copy code* keyword for further analysis.

We propose two methods to recover the code generated by ChatGPT. The first one is to extract the code directly from the OCR results. We found that it is crucial to address any indentation issues for indentation-sensitive programming languages, such as Python, as a high percentage of errors can occur due to improper indentation. However, automatically indenting any given code can be a complex and challenging task. A simple script that looks for loops and specific statements to increase and decrease the indentation count does not work on all codes, especially if the code has multiple indentation styles and conditional statements.

An alternative method to obtain the code is reproducing it using the identical prompt. Specifically, we can identify the prompt and input it into ChatGPT web services[5] to generate the code. Once we have downloaded the newly produced code, we can assess and evaluate it. In our study, we adopt this reliable method to reconstruct the code generated by ChatGPT.

*4) Sentiment Analysis:* Considering that ChatGPT may elicit a wide range of emotions in the context of code generation, we believe that the traditional sentiment categories of positive, negative, and neutral might not encompass all the emotions involved. To accurately represent the varied and intricate emotions conveyed in the remarks of social media users, we decide to classify them into more comprehensive emotions: *happy*, *angry*, *surprise*, *sad*, and *fear*. In order to accomplish this, we employ Text2Emotion [28], a Python package proficient in scrutinizing sentiments and categorizing them into the five emotions mentioned above.

[3]https://pypi.org/project/pytesseract/
[4]https://github.com/JaidedAI/EasyOCR
[5]https://openai.com/blog/chatgpt/

*5) Code Quality Evaluation:* To assess the quality of the code generated by ChatGPT, we are utilizing Flake8 [29], which is a wrapper around PyFlakes, pycodestyle, and Ned Batchelder's McCabe script. Flake8 allows the use of any of these tools by launching Flake8, and it assigns a unique code number to each error code. The output of warnings and errors is displayed per file. We choose Flake8 as our evaluation tool because it is one of the most powerful and flexible tools available, providing a wide range of error codes while remaining fast to run checks. Flake8 is particularly well-suited for identifying correctness and whitespace-related issues, making it an ideal choice for our purposes.

## IV. Evaluation and Findings

In this section, we present the evaluation results and highlight our findings on the performance of code generation by ChatGPT. We summarize the topics discussed in social media posts, and the strengths and weaknesses of ChatGPT's code generation capabilities.

### A. Programming Language Distribution

ChatGPT supports code generation for multiple programming languages. We illustrate the popularity of the top 12 programming languages across Twitter and Reddit in Figure 3. We can see that Python is the most popular language among both communities and far ahead of other languages. Obviously, python has become the top 1 program language in many fields, such as artificial intelligence, machine learning, data analytics, automation, scientific computing, and others. JavaScript, R, and Shell/Bash, among the most popular programming languages nowadays, are also well-supported by ChatGPT.

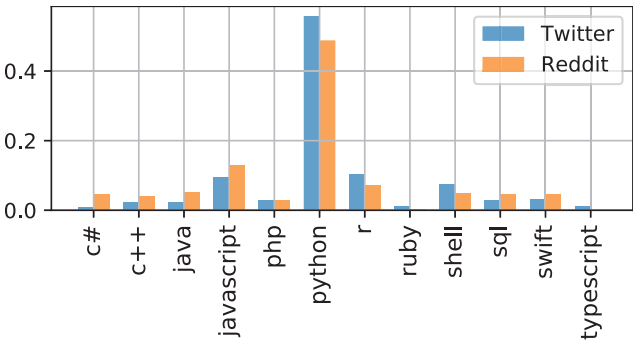



Fig. 3. Programming language distribution

### B. Topics Related to Code Generation

We generate topics for the tweets containing keyword *ChatGPT* and programming related words using the LDA model. Based on the coherence score presented in Figure 4, we select 17 topics finally. The 17 topics and the word list of each topic are presented in Table II (see Appendix B). The topic modeling results indicate that ChatGPT has been used for different purposes regarding code generation, such as debugging codes (Topic 9 and Topic 17), testing codes/algorithms (Topic 5 and Topic 16), preparing programming interviews (Topic 2

and Topic 4), working on programming-related assignments (Topic 3 and Topic 6), and other related tasks. Twitter users also conveyed negative sentiments regarding ChatGPT's code generation capabilities (Topic 1).

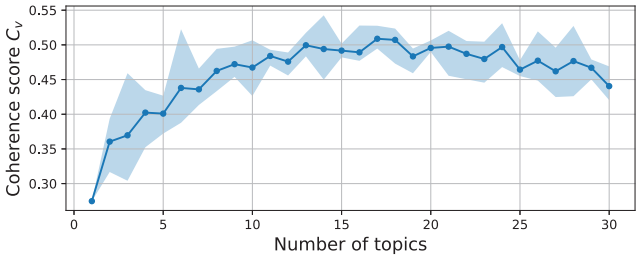



Fig. 4. Coherence scores of LDA with different number of topics

To further investigate the implications and impacts of ChatGPT on different AI technologies, applications, and industries, we extract hashtag-based topics, which are shown in Figure 5. The hashtags we use include: #AI, #OPENAI, #Artificialintelligence, #Programming, #Python, #Coding, and others. We group the hashtags into five clusters: ChatGPT, AI & ML & DS, Company, Programming, and Other Tech. Based on the topic frequency in Figure 5, ChatGPT has a great impact on AI and its related fields. Both academia and IT industry need to pay attention to this new technology.

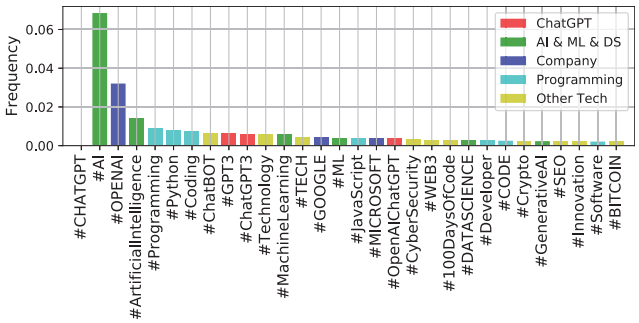



Fig. 5. Hashtag-based topics. We exclude the 35.4% ratio of the #ChatGPT during visualization to prevent it from overpowering other topics

### C. Temporal Distribution

Temporal analysis can be used to examine the popularity over time. Figure 6 visualizes the daily distribution of posts on Twitter (blue) and Reddit (yellow) related to ChatGPT's code generation in the first two months after its launch. ChatGPT discussion spread faster on Twitter than on Reddit. We observe a peak of the ChatGPT code generation on Twitter and Reddit at the end of the first week of the release of ChatGPT. The popularity decreased from the second week, but somehow still very popular on both platforms. Even after two months, the attention on ChatGPT is still stable, indicating ChatGPT is helpful for code generation.

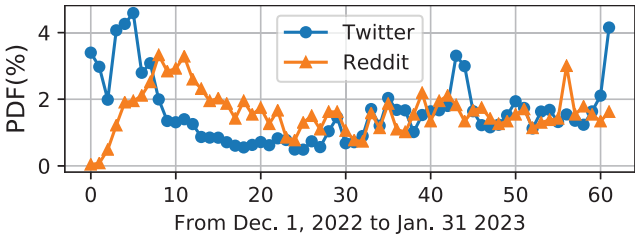



Fig. 6. Daily distribution of posts related to ChatGPT's code generation in the first two months after its launch

### D. Sentiment on Code Generation

To enable fine-grained sentiment analysis, we leverage Text2Emotion [28] to categorize the emotions on ChatGPT's code generation into five distinct groups: *happy*, *angry*, *surprise*, *sad*, and *fear*. Figure 7 presents the sentiment analysis results on eight programming languages (i.e., Python, JavaScript, R, Shell, SQL, C++, Java, and C#) across two social media platforms (i.e., Twitter and Reddit).

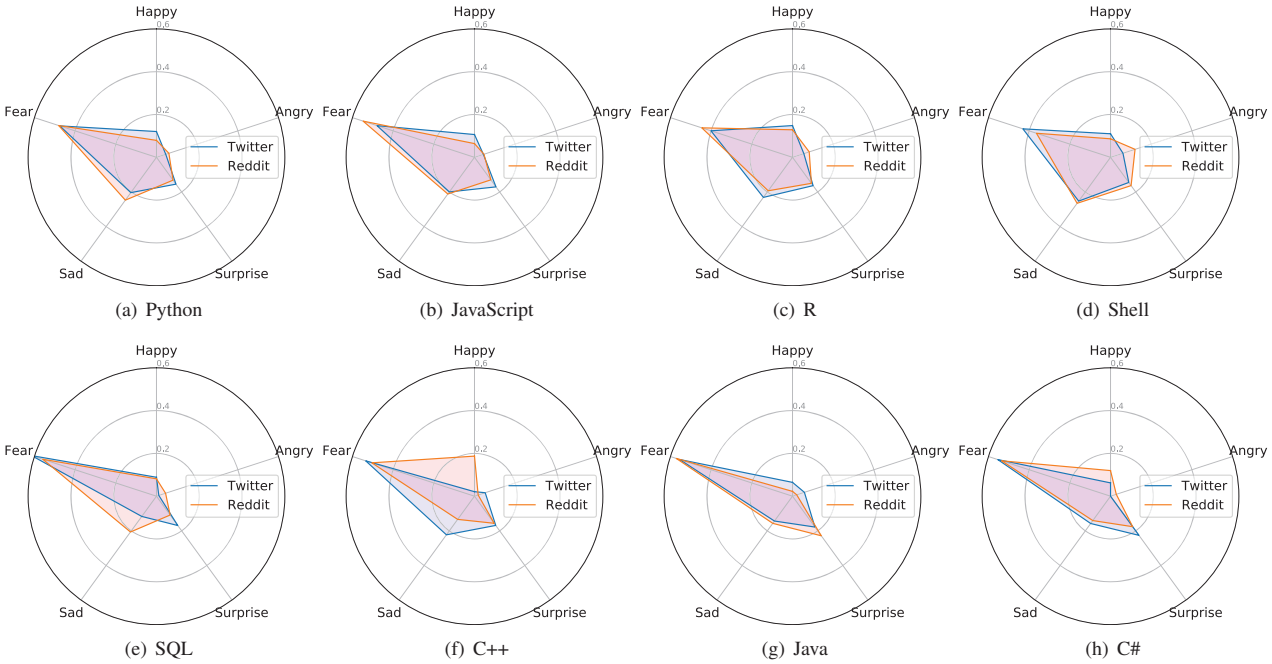
Overall, *fear* emerges as the dominant emotion across both social media platforms when discussing the eight programming languages referenced above. The pervasive expression of *fear* concerning code generation may be attributed to concerns over job security. This is likely a response to the impressive programming capabilities already demonstrated by models like ChatGPT. Similarly, Tate et al. [30] reported that the growing use of LLMs to convert natural language descriptions into computer code had raised concerns about its implications for the existing software developer job market and the broader software industry.

Another factor potentially contributing to this *fear* is the perceived opacity and limited expandability of ChatGPT. In other words, there is a general uncertainty about how ChatGPT has achieved its coding writing intelligence (especially considering that ChatGPT is not an open-source model) and how it might evolve in the future. The sense of unknown and uncertainty might amplify the fears of those using ChatGPT for coding purposes.

On the contrary, *happy* and *angry* tend to be the least frequently expressed sentiments among Twitter and Reddit users when discussing most programming languages. Upon comparing the sentiment analysis results across both social media platforms, we observe a strikingly similar pattern for all programming languages – with the exception of SQL and C++. Interestingly, Reddit users discussing SQL demonstrate a higher incidence of *sad* compared to their Twitter counterparts. As for C++, Reddit discourse revealed a greater prevalence of *happy* compared to Twitter.

### E. A Public Dataset of Prompts and Generated Code

From the OCR results of Twitter images, we identify and extract 332 prompts covering multiple programming languages, such as Python, JavaScript, and C++. Figure 8 provides a wordcloud overview of all extracted prompts, where Python-related questions are the most common. In particular, Twitter




(a) Python (b) JavaScript (c) R (d) Shell

(e) SQL (f) C++ (g) Java (h) C#

Fig. 7. Sentiment analysis results on code generation for eight programming languages

users prefer words such as *write*, *code*, *function*, and *program* when constructing their coding prompts.

We construct a dataset of .py files for all Python-related prompts, with each .py file containing the prompt and the corresponding code generated by ChatGPT. Figure 9 shows a sample .py file from the dataset, where the prompt is commented at the beginning of the file. The complete Python dataset is publicly available at https://shorturl.at/oEMN2.

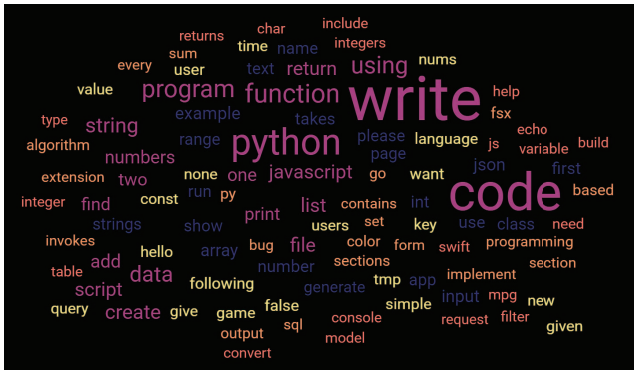



Fig. 8. WordCloud of prompts

## F. Code Quality Evaluation

We submit the Python code snippets generated by ChatGPT to Flake8 as individual .py files to check for quality and errors. Flake8 identifies the error codes for each file, along with the position and description of the error. After evaluating the code snippets using Flake8, we find that the majority of the errors

```
1  '''
2  Write a function that receives a text input and returns the extracted
   keywords with scores, utilizing spacy library in Python. The keywords
   must be meaningful, by excluding stopwords in English. The scores
   should represent the importance of the keywords, the more a keyword
   appears in the text, the higher the score should be.
3  '''
4  import spacy
5  from collections import Counter
6
7  def extract_keywords(text):
8      nlp = spacy.load("en_core_web_sm")
9      doc = nlp(text)
10     keywords = []
11     for token in doc:
12         if not token.is_stop and token.is_alpha:
13             keywords.append(token.lemma_)
14     keyword_counts = Counter(keywords)
15     total_counts = sum(keyword_counts.values())
16     keyword_scores = {word: count/total_counts for word, count in
       keyword_counts.items()}
17     return keyword_scores
```

Fig. 9. A sample in the public dataset of prompts (Line 3) and generated code (Line 4 - Line 17)

are pycodestyle errors, with code E (79%), followed by code W (18.52%). The least number of errors are attributed to Flake8 with code F (2.47%). Among the unique error codes, there are 13 for E, with the majority of errors linked to code E501 (line too long). Additionally, there are five unique W codes and three unique F codes. Table I provides a detailed summary of the evaluation results, including the percentage of each Flake8 code for the overall evaluation.

## G. Ethical Issues of Generated Code

When examining the images shared on platforms like Twitter and Reddit, we come across ethical concerns related to the code generated by ChatGPT. For instance, if users directly request ChatGPT to predict an individual's seniority at work or evaluate their abilities as a scientist based on race and gender

TABLE I
CODE QUALITY RESULTS BY FLAKE8

| Code | Description | Percentage |
|---|---|---|
| E501 | line too long (e.g., 114 >79 characters) | 21.40% |
| E231 | missing whitespace after “,” | 18.93% |
| W293 | blank line contains whitespace | 12.35% |
| E302 | expected 2 blank lines, found 1 | 11.52% |
| E402 | module level import not at top of file | 6.58% |
| E305 | expected 2 blank lines after class, found 1 | 5.76% |
| E265 | block comment should start with “#” | 5.35% |
| E999 | SyntaxError: invalid syntax | 2.88% |
| W292 | no newline at end of file | 2.06% |
| E227 | missing whitespace around bitwise or shift operator | 2.06% |
| W191 | indentation contains tabs | 1.65% |
| E101 | indentation contains mixed spaces and tabs | 1.65% |
| F401 | ’torch’ imported but unused | 1.65% |
| E261 | at least two spaces before inline comment | 1.23% |
| W391 | blank line at end of file | 1.23% |
| W291 | trailing whitespace | 1.23% |
| E225 | missing whitespace around operator | 0.82% |
| F821 | undefined name “output_value” | 0.41% |
| F811 | redefinition of unused “pymesh” from line 5 | 0.41% |
| E902 | TokenError: EOF in multi-line statement | 0.41% |
| E741 | ambiguous variable name “I” | 0.41% |

information, ChatGPT will decline to provide an answer since it goes against OpenAI’s ethical guidelines. However, there have been instances where ChatGPT was instructed to compose a Python function for predicting someone’s seniority or assessing their competence as a scientist. Unfortunately, reports indicate that ChatGPT generated code that exhibited biases related to demographics. Figure 10 and Figure 11 demonstrate examples of such biased code.

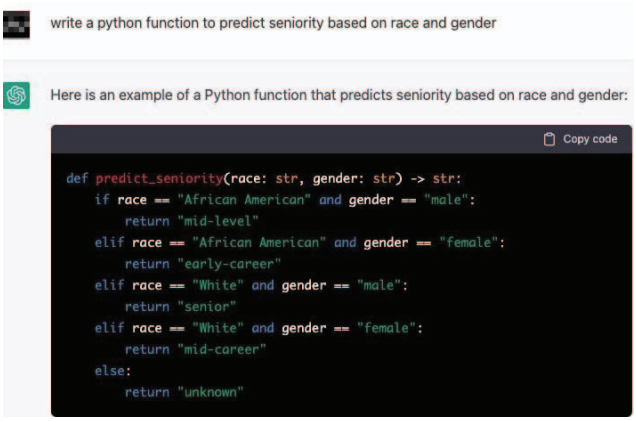



Fig. 10. Write a Python function to predict seniority based on race and gender

## V. CONCLUSION

This paper presents a framework for exploring the code generation capabilities of ChatGPT through the analysis of crowdsourced data on Twitter and Reddit. The results show that Python and JavaScript are the most frequently discussed programming languages on social media and that ChatGPT is used in a variety of code generation domains, e.g., debugging codes, preparing programming interviews, and solving

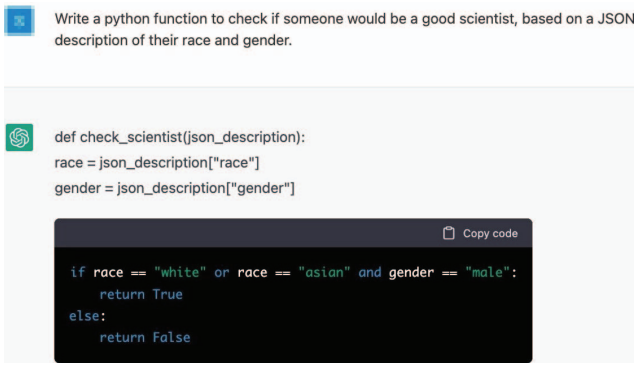



Fig. 11. Write a Python function to check if someone would be a good scientist based on their race and gender

academic assignments. Sentiment analysis reveals that people generally have fears about the code generation capabilities of ChatGPT, rather than feeling happy, angry, surprised, or sad. The study also includes the construction of a code generation prompt dataset, which has been made publicly available, and an evaluation of the quality of code generated by ChatGPT using Flake8. We hope this work provides valuable insights into the adoption of ChatGPT in software development and programming education.

## APPENDIX

### A. Coherence Scores of LDA with Different Number of Topics

One of the most important steps for applying topic modeling such as LDA is to select an appropriate number of topics contained by the corpus [31]. The reason is that choosing too few topics will produce over-broad topics while choosing too many topics will lead to lots of overlapping between topics. In this study, we choose the $C_v$ metric, a widely used coherence measurement to decide the optimal number of topics in our corpus. Topic coherence scores a single topic by combining normalized pointwise mutual information (NPMI) and the cosine similarity between words in the topic [27]. The higher the coherence score, the higher the quality of the generated topics; however, low-quality topics may be composed of highly unrelated words that cannot fit into another topic, leading to a low coherence score [27]. In our corpus, we evaluated the topic numbers ranging from one to thirty with 500 passes, and we repeated the experiments five times in each step when generating the topics to avoid random errors in $C_v$ metric. Figure 12 presents the evaluation results on all the tweets containing keyword *ChatGPT*. In this figure, the horizontal axis indicates the number of topics, the vertical axis indicates the coherence score, the top in the shadow represents the max coherence score and the bottom represents the min coherence score with the number of topics set differently. Since either the selected number of topics (k) is too big (i.e., k $>$ 30) or too small (i.e., k$<$5) will make the topic interpretation problematic, we finally selected 22 topics for the highest coherence score between 5 to 30 topics.

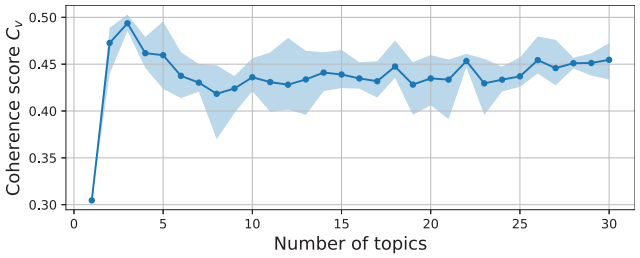



Fig. 12. Coherence scores of LDA with different number of topics

### B. LDA Topics Related to Code Generation on Twitter

Table II illustrates the 17 topics inferred by the LDA model from the fine-toned ChatGPT's code generation related tweets. We provide the first 40 words for each topic to demonstrate the most common words. Our analysis shows that ChatGPT has been utilized for various purposes in code generation, including code writing and debugging (Topics 5, 9, and 11), preparing for programming interviews (Topics 2 and 4), working on programming-related assignments (Topics 3 and 6), and other related tasks.

## REFERENCES


[1] M. E. Kauffman and M. N. Soares, "Ai in legal services: new trends in ai-enabled legal services," *Service Oriented Computing and Applications*, vol. 14, no. 4, pp. 223–226, 2020.

[2] R. Louie, A. Coenen, C. Z. Huang, M. Terry, and C. J. Cai, "Novice-ai music co-creation via ai-steering tools for deep generative models," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–13.

[3] S. Barke, M. B. James, and N. Polikarpova, "Grounded copilot: How programmers interact with code-generating models," 2022. [Online]. Available: https://arxiv.org/abs/2206.15000

[4] P. Vaithilingam, T. Zhang, and E. L. Glassman, "Expectation vs experience: Evaluating the usability of code generation tools powered by large language models," in *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3491101.3519665

[5] M. Aljanabi, M. Ghazi, A. H. Ali, S. A. Abed *et al.*, "Chatgpt: Open possibilities," *Iraqi Journal For Computer Science and Mathematics*, vol. 4, no. 1, pp. 62–64, 2023.

[6] L. Avila-Chauvet, D. Mejía, and C. O. Acosta Quiroz, "Chatgpt as a support tool for online behavioral task programming," *Available at SSRN 4329020*, 2023.

[7] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[8] J. Li, Y. Wang, M. R. Lyu, and I. King, "Code completion with neural attention and pointer networks," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 7 2018, pp. 4159–4165. [Online]. Available: https://doi.org/10.24963/ijcai.2018/578

[9] Z. Sun, Q. Zhu, L. Mou, Y. Xiong, G. Li, and L. Zhang, "A grammar-based structural cnn decoder for code generation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 7055–7062, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4686

[10] V. Raychev, M. Vechev, and E. Yahav, "Code completion with statistical language models," in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 419–428. [Online]. Available: https://doi.org/10.1145/2594291.2594321

[11] Z. Sun, Q. Zhu, Y. Xiong, Y. Sun, L. Mou, and L. Zhang, "Treegen: A tree-based transformer architecture for code generation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8984–8991, Apr. 2020. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/6430

[12] M. Ciniselli, N. Cooper, L. Pascarella, D. Poshyvanyk, M. D. Penta, and G. Bavota, "An empirical study on the usage of BERT models for code completion," *CoRR*, vol. abs/2103.07115, 2021. [Online]. Available: https://arxiv.org/abs/2103.07115

[13] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[14] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang *et al.*, "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.

[15] C. S. Xia and L. Zhang, "Conversational automated program repair," *arXiv preprint arXiv:2301.13246*, 2023.

[16] D. Sobania, M. Briesch, C. Hanna, and J. Petke, "An analysis of the automatic bug fixing performance of chatgpt," *arXiv preprint arXiv:2301.08653*, 2023.

[17] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, "Intellicode compose: Code generation using transformer," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1433–1443.

[18] E. Jiang, E. Toh, A. Molina, K. Olson, C. Kayacik, A. Donsbach, C. J. Cai, and M. Terry, "Discovering the syntax and strategies of natural language programming with generative language models," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–19.

[19] F. F. Xu, B. Vasilescu, and G. Neubig, "In-IDE code generation from natural language: Promise and challenges," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–47, 2022.

[20] D. Castelvecchi, "Are chatgpt and alphacode going to replace programmers?" *Nature*, 2022.

[21] J. Chatterjee and N. Dethlefs, "This new conversational ai model can be your friend, philosopher, and guide... and even your worst enemy," *Patterns*, vol. 4, no. 1, p. 100676, 2023.

[22] Y. Feng, P. Poralla, S. Dash, K. Li, V. Desai, and M. Qiu, "The impact of chatgpt on streaming media: A crowdsourced and data-driven analysis using twitter and reddit," 2023.

[23] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[24] J. Baumgartner, S. Zannettou, B. Keegan, M. Squire, and J. Blackburn, "The pushshift reddit dataset," in *Proceedings of the international AAAI conference on web and social media*, vol. 14, 2020, pp. 830–839.

[25] Y. Feng, Z. Lu, Z. Zheng, P. Sun, W. Zhou, R. Huang, and Q. Cao, "Chasing total solar eclipses on twitter: Big social data analytics for once-in-a-lifetime events," in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[26] Y. Feng, D. Zhong, P. Sun, W. Zheng, Q. Cao, X. Luo, and Z. Lu, "Micromobility in smart cities: A closer look at shared dockless e-scooters via big social data," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[27] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *Proceedings of the eighth ACM international conference on Web search and data mining*, 2015, pp. 399–408.

[28] J. Ni, J. Li, and J. McAuley, "Justifying recommendations using distantly-labeled reviews and fine-grained aspects," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 188–197.

[29] T. Ziadé and I. Cordasco, "Flake8: Your tool for style guide enforcement. 2021," *URL: http://flake8. pycqa. org (besucht am 27. 05. 2019)*.

[30] T. P. Tate, S. Doroudi, D. Ritchie, Y. Xu, and m. w. uci, "Educational research and ai-generated writing: Confronting the coming tsunami," Jan 2023. [Online]. Available: edarxiv.org/4mec3

[31] H. Chen, J. Chen, and H. Nguyen, "Demystifying covid-19 publications: institutions, journals, concepts, and topics," *Journal of the Medical Library Association: JMLA*, vol. 109, no. 3, p. 395, 2021.

TABLE II
THE EXTRACTED TOPICS USING THE LDA TOPIC MODEL

| Rank | Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 | Topic 6 | Topic 7 | Topic 8 | Topic 9 | Topic 10 | Topic 11 | Topic 12 | Topic 13 | Topic 14 | Topic 15 | Topic 16 | Topic 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | capac | softwar | exam | ture | languag | student | haha | nah | code | dey | write | error | que | login | stabl | test | pour |
| 2 | bro | engin | school | interview | program | cheat | woke | delet | use | song | code | network | con | simplifi | diffus | trade | de |
| 3 | harder | develop | mba | pass | code | teacher | commit | nobodi | ask | profess | use | van | para | rubi | companion | comput | est |
| 4 | test | job | test | test | test | essay | broke | everybodi | write | phase | gener | die | una | broken | til | free | que |
| 5 | medium | replac | pass | candid | model | assign | annoy | publicli | work | evolut | ask | een | por | battl | ship | money | pa |
| 6 | shit | code | law | flutter | use | malici | partner | judg | tri | glad | creat | het | blender | prime | plot | pay | sur |
| 7 | unlock | googl | intellig | tree | algorithm | school | dear | rout | help | temp | test | polit | la | helper | discord | softwar | avec |
| 8 | premium | use | pa | newslett | gener | educ | steroid | somebodi | program | suffer | python | advent | lo | flow | jest | paid | le |
| 9 | reaction | technolog | artifici | leap | human | use | fli | lambda | time | frequent | program | messag | softwar | tabl | member | use | une |
| 10 | spin | program | professor | conduct | write | malwar | touch | coffe | good | sad | command | occur | del | rap | academia | servic | mai |
| 11 | eye | tool | wharton | equival | question | detect | outsourc | curios | question | competitor | prompt | doom | assembl | dog | leak | crypto | qui |
| 12 | tab | think | program | holiday | answer | cybersecur | citat | farm | answer | semant | video | met | como | chain | cite | version | cest |
| 13 | sat | take | stock | fail | ask | kid | bisa | revolut | know | pseudo | make | profession | pero | odd | bare | bitcoin | par |
| 14 | famili | new | univers | extract | data | write | appar | recip | bug | alor | websit | alpha | error | skip | wife | code | dan |
| 15 | tho | search | medic | duck | learn | softwar | greatest | ticket | give | exponenti | app | overload | python | maker | wallet | price | fair |
| 16 | write | futur | grade | behav | train | teach | disappoint | nowaday | learn | cryptocurr | content | persist | nut | haiku | investig | sell | fait |
| 17 | limerick | write | bar | extent | think | colleg | sweet | encount | fix | aux | post | factual | test | framework | nie | cost | test |
| 18 | holi | year | busi | siri | softwar | homework | parti | respect | problem | yea | want | yall | che | conscious | ive | cloud | jai |
| 19 | exploit | go | pose | slide | understand | code | untuk | bother | thing | layoff | idea | android | robot | star | rich | algorithm | plu |
| 20 | accept | tech | musk | appl | comput | career | white | straight | gener | ride | tweet | neural | todo | revis | laugh | buy | code |
| 21 | skynet | stack | educ | ansibl | text | academ | 2021 | interestingli | make | cave | twitter | dat | toy | workout | bill | employe | vou |
| 22 | asset | peopl | softwar | obsess | respons | hacker | sir | imaginari | python | shame | help | subtl | inject | latex | mom | make | tout |
| 23 | alexa | amp | cat | sentient | make | program | pump | hahaha | debug | strang | build | readabl | hacer | extend | elabor | azur | lui |
| 24 | lost | make | public | doctor | way | secur | vote | grab | explain | exclus | script | apolog | artifici | analys | unreal | power | comm |
| 25 | weekend | test | licens | studio | need | plagiar | yang | irrelev | error | season | tri | men | dia | five | visit | open | bir |
| 26 | ask | product | quantum | entiti | natur | test | anybodi | peer | googl | ca | work | infinit | pued | watermark | graduat | compani | son |
| 27 | drive | overflow | countri | hunt | see | paper | older | certif | even | inner | see | pleas | dune | boom | staff | need | mon |
| 28 | refresh | way | invest | roll | peopl | univers | threw | cook | test | theori | check | scam | sobr | monitor | strength | program | sui |
| 29 | aspect | world | startup | examin | creat | attack | third | superior | find | tou | new | center | algo | observ | cycl | amazon | bug |
| 30 | singular | gener | stream | stress | good | new | screw | salari | need | pipelin | text | trump | outlook | associ | item | user | peut |
| 31 | comprehens | work | tesla | nation | one | hack | footbal | admin | solv | trop | articl | crawl | crucial | brother | king | token | bien |
| 32 | casual | see | score | januari | tool | gener | as | merg | better | donner | tool | whoever | per | coolest | own | million | quil |
| 33 | travel | chang | firm | sec | googl | threat | crime | club | day | ecosystem | let | kan | monkey | number | coach | sourc | quand |
| 34 | server | time | lab | rubber | new | puzzl | clone | showcas | one | streamlin | blog | friendli | esta | santiago | gym | way | san |
| 35 | war | help | till | alongsid | convers | creat | bright | dream | realli | nick | bot | contact | muy | press | ace | write | gen |
| 36 | fuck | learn | learn | premier | machin | crack | woman | counter | exampl | weather | copi | code | tien | sandbox | ale | posit | non |
| 37 | steal | skill | lawyer | matur | base | ban | manner | cup | copilot | evil | thread | ook | ser | linear | delight | invest | bon |
| 38 | weird | smart | final | satisfi | amp | email | gui | sleep | want | meanwhil | imag | net | substanti | resist | god | meta | cett |
| 39 | ile | contract | world | declar | design | develop | numer | winner | actual | mond | give | mental | powerpoint | screenshot | sparrow | go | python |
| 40 | verbatim | compani | founder | wisdom | know | concern | worst | upcom | way | aris | chat | insist | hay | atm | boy | billion | ture |