EasyChair Preprint

№ 15569

Evaluating LLMs for Arabic Code Summarization: Challenges and Insights from GPT-4

Ahmed Aljohani, Raed Alharbi, Asma Alkhaldi and Wajdi Aljedaani

December 13, 2024

Evaluating LLMs for Arabic Code Summarization: Challenges and Insights from GPT-4

Ahmed Aljohani*, Raed Alharbi†, Asma Alkhaldi‡, and Wajdi Aljedaani*
*University of North Texas, †Saudi Electronic University, ‡Saudi Data and Artificial Intelligence Authority
{ahmed.aljohani, wajdi.aljedaani}@unt.edu, ri.alharbi@seu.edu.sa, aakhaldi@sdaia.gov.sa

***Abstract*—GPT-4 —the backbone of ChatGPT—has demonstrated remarkable performance in both natural language and source code tasks. Recently, Large Language Models (LLMs) like GPT-4 have significantly advanced software engineering tasks such as code summarization. These advancements boost developer productivity and help address often neglected tasks like code documentation. While code summarization and commenting are essential for maintaining code quality and facilitating communication among developers, writing comments manually is time-consuming. Although several studies have proposed and evaluated deep learning-based approaches and LLMs to automate comment generation, these efforts primarily focus on the English language, leaving a gap for other languages, particularly Arabic. In this study, we evaluate the ability of GPT-4 to generate accurate Arabic comments. We support our evaluation with both manual and automatic analysis to measure the correctness and nature of the generated comments. Our findings reveal that while GPT-4 generally produces correct Arabic summaries, they often do not align with the developer's intent as reflected in the BERT-Similarity, ROUGE and BLEU scores. We also show that GPT-4's comments are more verbose due to the morphological richness of the Arabic language and a systematic approach that tends to describe each code component in detail. Finally, the readability of these comments is moderate, with scores ranging from 30.29 to 100.**

***Index Terms*—GPT-4, LLMs, Code Summarization, Arabic Language.**

I. INTRODUCTION

Code summarization – known as comments [29], serves as an essential bridge between human understanding and machine-executable code. It provides context, rationale, and explanations that are often not immediately obvious from the code itself [19], [28]. Code comments enhance the readability and maintainability of software by offering insights into the developer's intent, the functionality of complex algorithms, and potential pitfalls or limitations within the code [7], which can significantly reduce the time and effort required to debug, extend, or refactor software. While source code comments are crucial, writing them is often time-consuming and can be easily neglected by developers [24]. The effort required to create clear and brief comments, especially under tight deadlines, leads to prioritizing coding over documentation.

Several studies [11], [17], [1] have proposed automatic code summarization techniques based on deep learning-based to facilitate comment generation. These techniques utilize powerful generative models trained on large-scale code-comment datasets to translate code snippets into natural language summaries. Building on the success of pre-training and fine-tuning

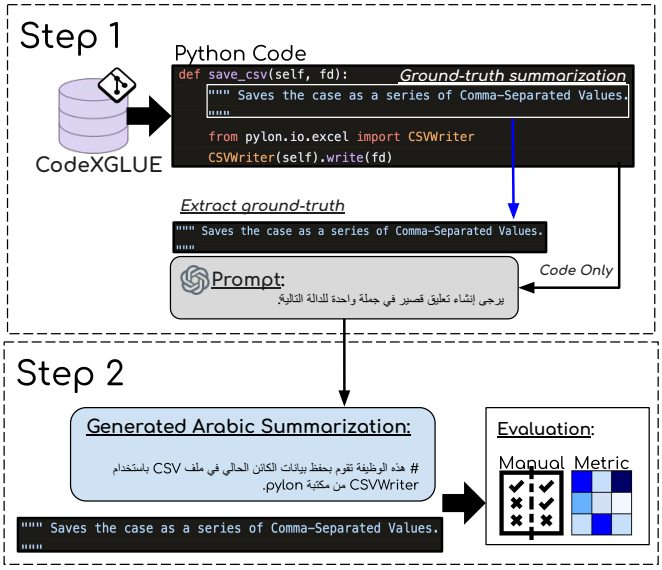

Figure 1: Overall Approach

paradigms, code models like CodeBERT and CodeT5 [9], [27] have been introduced to further enhance code-related tasks, including code summarization. These models are first pre-trained on general language tasks and then fine-tuned for code-related tasks such as code summarization.

Recent research has also shown promising results of Large Language Models (LLMs) like ChatGPT—based on GPT-3.5 and GPT-4—on code-related tasks [6], [15], [5]. For instance, Sun et al. [24] demonstrated that ChatGPT can generate brief and accurate code summarizations for a given prompt and code snippets. The work highlights that ChatGPT is capable of producing detailed and semantically rich comments compared to models like CodeBERT and CodeT5 in traditional evaluation metrics such as ROUGE-L and BLEU. Despite the significant advancements in these techniques, English remains the primary language used for automatically generating code comments.

Source code comments are not limited to English; languages such as Chinese and Arabic are also prevalent in codebases [18]. However, efforts to study and evaluate code comments in right-to-left languages, particularly Arabic, are significantly less common. For example, a recent comprehensive evaluation of ChatGPT on Arabic NLP tasks by Khondaker et al. [13] highlighted the model's performance across a wide range of

Arabic language tasks, from natural language understanding to text generation. Despite this extensive analysis and ChatGPT's high performance in Arabic tasks, the study did not assess ChatGPT's capabilities in the specific domain of Arabic code summarization.

To the best of our knowledge, there are no existing models or evaluation frameworks specifically designed to support or assess the automatic generation of Arabic comments. LLMs like GPT-4 present a promising solution for Arabic-speaking developers. However, such a model has not yet been evaluated for its effectiveness and accuracy in generating Arabic comments. We aim to fill this gap by investigating the ability of GPT-4 to produce accurate and contextually correct code comments in Arabic. The following research questions guide our assessment:

RQ1: How accurate is GPT-4 in generating Arabic code summarizations?

Motivation: Despite advancements in LLMs such as GPT-4, their evaluation of Arabic code-related tasks remains limited. This study addresses this gap by assessing the functional accuracy of LLMs in Arabic code summarization.

RQ2: What is the nature of the Arabic summarizations/-comments generated by GPT-4?

Motivation: While accuracy is essential, understanding the nature of the generated text—its length, style, e.g., Part of Speech, length, and readability—is crucial. This analysis will show how well GPT-4 incorporates Arabic context and coding language.

Our findings for RQ1 indicate that the GPT-4 model predominantly generates accurate Arabic comments, with a few instances being partially incorrect or completely inaccurate. We also observed that GPT-4 tends to explain the individual components of the code within the comments. While this is systematically correct, the model often fails to capture the abstract intention of the code's functionality. Regarding RQ2, the generated Arabic comments tend to be longer than the ground-truth, with the majority starting with a noun. Additionally, the readability score of these generated comments suggests they are fairly easy to read.

II. RELATED WORK

Code summarization is the process of generating clear and informative descriptions for pieces of source code [29]. Several methods, including LLMs, were driven towards automatic code summarization to support developers [11], [17], [1], [2].

Most recently, Sun in [24] investigated the performance of ChatGPT in the context of automatic code summarization, specifically focusing on Python code snippets from the CSN-Python dataset. The authors evaluated ChatGPT against three code summarization models: NCS, CodeBERT, and CodeT5. The study used three widely recognized metrics—BLEU, METEOR, and ROUGE-L—to assess the quality of the generated summaries. The findings reveal that while ChatGPT is capable of generating detailed and semantically rich comments, its performance in terms of BLEU and ROUGE-L is lower than that of the SOTA models. ChatGPT does perform comparably to CodeT5 in the METEOR metric, but overall, it falls short in comparison to the other models.

Despite the effectiveness of the proposed evaluation for code summarization using LLMs—ChatGPT, the focus has primarily been on evaluating code summarization in English. Thus, our study used a similar approach, but we applied it to the scope of Arabic code summarization. Along this direction, we extend our evaluation beyond standard techniques like automatic scores (e.g., BLEU and ROUGE-L), incorporating manual analysis and investigating the nature of the generated comments. To the best of our knowledge, this is the first study to address the evaluation of code summarization in the Arabic language.

III. STUDY DESIGN

This section describes our methodology for evaluating the accuracy and functional correctness of Arabic comments generated by GPT-4. Additionally, we examine the nature of the Arabic in the code summarization task. Figure 1 presents an overview of our approach. We designed our input to include the Arabic prompt and the Python code. The model generates the Arabic comment for the giving input to be evaluated manually and automatically.

A. Dataset

Following similar studies [4], [3], [2], we used the CodeXGLUE dataset [16] to benchmark code-related tasks. The dataset includes around 280K Python code snippets with developer-written comments. For our manual evaluation of Arabic comments, we selected a statistically significant sample of 384 Python methods (95% confidence level, 5% margin of error) and limited the code snippets to a maximum of 10 lines. This selection ensured a manageable sample size and non-complex code for accurate manual evaluation. We focused on Python due to its popularity[1] and the natural language quality of its ground-truth comments (i.e., they are free from documentation tags like `@param` and `@return`), which simplifies both manual and automatic evaluation. Table I summarizes the distribution of line-of-code (LOC) and comment length in our sample.

Table I: Statistics on Lines of Code and Words per Comment

	Line of Code (LOC)			Words per Comment		
	Average	Max	Min	Average	Max	Min
Values	4.73	10	2	8.61	46	2

B. Prompts

Despite the availability of advanced prompt techniques [21], we opted for simple, precise, and clear prompts in our study. While using more sophisticated prompt techniques might yield better code summarizations, our goal was to create a prompt that is practical and realistic. We adopted a similar approach to

[1]Stack Overflow Developer Survey 2023: https://survey.stackoverflow.co/2023/; GitHub State of Open Source and AI 2023: https://github.blog/2023-11-08-the-state-of-open-source-and-ai/

[24] in constructing our prompt but tailored it specifically for the Arabic language, as shown below:

> يرجى إنشاء تعليق قصير في جملة واحدة للدالة التالية:
> <Python code>

C. LLM models - GPT4

To generate Arabic summaries for source code, we selected GPT-4 due to its high performance in both Arabic and code-related tasks. GPT-4 has outperformed other LLMs in the HumanEvalPlus benchmark [14] and has been widely adopted in research and applications [6], [15], [22]. Additionally, it has demonstrated strong results in Arabic-language tasks [25], [13]. For our experiment, we used the `gpt-4-turbo`[2] API provided by OpenAI.

D. Evaluation

We conducted two types of evaluation. The first type is *manual*, where the generated Arabic comments are labeled (0 or 1) based on predefined categories (Correct, Partially Correct, Incorrect). This manual analysis provides an initial understanding of GPT-4' capabilities in generating Arabic code summaries. The second type of evaluation, which is more commonly used in the literature, is *automatic* evaluation [20]. For this type of evaluation, we employed the following metrics: ROUGE, BLEU, and Multilingual Similarities.

1) Manual Evaluation: Current code summarization benchmarks consist of code snippets paired with English-written comments. This facilitates the automatic evaluation (i.e., ROUGE, BLEU) of the generated summaries by LLMs, as both the generated and original comments are in the same language (English-to-English). Unlike these existing benchmarks [16], [12], which are predominantly in English, there is no established benchmark for code summarization in Arabic. Thus, we conducted a manual analysis of generated comments in Arabic. In our manual evaluation process, we categorized the generated Arabic comments into three distinct labels driven by this paper [17]:

- **Correct:** The summary accurately summarizes or describes the method's functionality.
- **Partially Correct:** The summary is missing or has information, which could change the understanding of the method's functionality.
- **Incorrect:** The summary incorrectly describes the functionality.

The manual process is accomplished independently by two researchers who are familiar with both the Arabic and Python languages and a third inspector to resolve the labeling mismatch. Cohen's Kappa coefficient was then applied to assess the level of agreement between the inspectors and the researcher, yielding a score of **0.87**, which is considered substantial [10].

2) Automatic Evaluation: To support our manual analysis, we computed two n-gram and overlap-based NLP metrics, Bilingual Evaluation Understudy (BLEU) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [23], as well as an embedding-based similarity metric, BERT-Similarity using the Multilingual E5 model [26].

Given that the generated comments are in Arabic and the ground-truth comments are in English, BLEU and ROUGE would not be effective since these metrics require both texts to be in the same language. Therefore, we automatically translated the generated Arabic comments into English before applying the BLEU and ROUGE metrics, ensuring the translation was accurate and did not introduce any additional information. To further support the automatic evaluation, we employed BERT-Similarity, which can assess the similarity between texts in different languages, such as Arabic and English. For this, we first used the Multilingual E5 model [26] to extract text embeddings and measure the semantic similarity between the ground-truth and the generated summaries.

E. Data Preprocessing

The generated Arabic summarizations include symbols (e.g., #) and non-Arabic text, such as Russian, that could affect the length, readability, and POS analysis. To ensure accurate measurements, we used **Stanza**[3], an NLP library from Stanford, to systematically clean the comments by removing irrelevant information. However, we retained the English text, as it is a crucial part of the comments that describe variables or identifiers.

F. Arabic Readability Metrics

We employed the Osman Arabic Readability Index [8] to evaluate the clarity and readability of the generated Arabic code summaries. The Osman metric is a comprehensive formula specifically designed for the Arabic language. It takes into account linguistic features unique to Arabic, which are not typically captured by traditional readability metrics. The formula considers various components, as shown below:

$$\text{Osman} = 200.791 - 1.015\left(\frac{A}{B}\right) - 24.181\left(\frac{C+D+G+H}{A}\right) - \text{English Penalty}$$

Where:

- A is the total number of words,
- B is the total number of sentences,
- C is the number of hard words (words with more than 5 letters),
- D is the total number of syllables in the text,
- G is the number of complex words (words with more than four syllables),
- H is the number of "Faseeh" words,
- English Penalty is calculated based on the proportion of English words relative to Arabic words, scaled by a factor

[2]https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

[3]https://stanfordnlp.github.io/stanza/available_models.html

Table II: Sample of generated Arabic summarization.

No.	Code Snippet	Ground-Truth	Arabic Comment	Label

1	`def _pp(dict_data):`
	`  for key, val in dict_data.items():`
	`  print('{0:<11}:{1}'.format(key, val))`
		Pretty print.	# تعليق يصف الوظيفة: هذه الوظيفة تقوم بطباعة مفاتيح وقيم القاموس المُعطى بتنسيق محدد حيث تكون المفاتيح محاذاة لليسار ومتبوعة بالقيم التوافقة معها	Correct

2	`def bulk_copy(self, ids):`
	`  schema = DeviceSchema()`
	`  return self.service.bulk_copy(self.base, self.RESOURCE, ids, schema)`
		Bulk copy a set of devices.	# تقوم هذه الدالة بنسخ مجموعة من الأجهزة بناءً على قائمة الأيدي المدخلة
			использать глагол в том (past, present, future, etc.) и в форме, которая подходит контексту. , .
				Partially Correct

3	`def format(file_metrics, build_metrics):`
	`  metrics = {'files': file_metrics}`
	`  if build_metrics:`
	`  metrics['build'] = build_metrics`
	`  body = json.dumps(metrics, sort_keys=True, indent=4) + '\n'`
	`  return body`
		Compute output in JSON format.	# هذه الوظيفة تقوم بحذف معطيات الملفات وبيانات البناء إلى صيغة JSON.	Incorrect

of 5 that represents the maximum impact these words have on the readability score.

Inclusion of English Penalty: A unique aspect of our approach was the addition of an "English Penalty" factor, which was not included in the original Osman metric. This modification was necessary due to the presence of English words in some generated summaries (e.g., Table II No.3), which can significantly affect the readability for Arabic-speaking developers. To quantify this impact, we introduced a scaled penalty with a factor of 5 (i.e., an adjustable threshold). This scaling ensures that the presence of English words is penalized proportionately.

IV. STUDY RESULTS

In RQ1, we present the results of the manual and automatic analysis supported by examples generated by the GPT-4 model. The answer to RQ2 presents the nature of the generated Arabic summarization of the code, which includes the length of the Arabic comments, their Part-Of-Speech (POS), and their readability.

A. *RQ1: How accurate are LLMs in generating Arabic code summarizations?*

Manual Analysis: The manual evaluation of the generated Arabic comments shows a high degree of accuracy, with 97.14% being functionally correct. Additionally, 2.34% of the comments were found to be partially correct, indicating a minor deviation from the desired outcomes yet retaining some functional relevance. Finally, only one comment, representing 0.52% of the total, i.e., 384 samples, was categorized as incorrect.

Table II provides examples from the three labels (correct, partially correct, and incorrect) along with the code snippet and the ground truth. From the same table, example No.1 is an instance of *correct* Arabic comments, where GPT-4 successfully captured the functionality of the code comprehensively without deleting crucial details that might affect its utility. However, our analysis shows that the generated Arabic comments were highly systematic, where the model tended to describe each code component in detail. This includes specific actions like outputting text through the print method, iterating with loops, and accessing dictionary keys and items. Consequently, this systematic approach led to comments that diverged from the developer's intended description. The developer's primary goal was to enhance the aesthetic layout of printed outputs ("Pretty print"), yet GPT-4's comments focused on describing technical actions. This disparity highlights a limitation in the model's ability to interpret and reflect higher-level, abstract functionalities in its generated comments.

Example No.2 illustrates a case of *partially correct* Arabic summarization. In this instance, the Arabic comment accurately reflects the systematic processes of the code. However, the comment's clarity is compromised by several factors. Firstly, the inclusion of Russian language terms within the Arabic comment introduces a linguistic inconsistency that can confuse the reader. Secondly, the model's handling of English technical terms, such as "id", also presents challenges. Specifically, the model translates "id" to "الأيدي", which is an English pronunciation but written in Arabic language. The word is more commonly means "hands" in Arabic rather than the intended "ids". A more appropriate Arabic technical term would be "معرف" or "هوية". Such a translation could potentially mislead developers, especially upon initial review, due to the significant semantic difference.

Example No.3 represents a case of *incorrect* Arabic summarization, where a critical error in verb usage significantly alters the perceived functionality of the code. The Arabic comment incorrectly states that the code ("بحذف") (delete) the data from the input file. In reality, the code's primary function is to organize and format the provided metrics data into a structured, readable JSON format. This misinterpretation likely stems from the systematic description approach used by the model, similar to that observed in Example No.1. We hypothesize that the use of the `json.dumps` function within the code may have been misinterpreted by the model as translating `dumps` to ("بحذف") (delete).

Table III: Metrics of Arabic Comments

BERT-Sim	BLEU1	BLEU2	ROUGE1	ROUGE2	ROUGE-L
0.55	0.15	0.06	0.23	0.06	0.19

Automatic Analysis: In Table III, The BERT-Similarity score of 0.55 suggests that there is only a moderate level of semantic similarity between the generated Arabic comments (translated into English) and the original English ground-truth comments. This moderate score indicates that while GPT-4 captures some aspects of the code's functionality, it often misses the intended meaning or description provided by the developer.

The low scores in BLEU and ROUGE metrics further highlight the limitations of the generated summaries. The BLEU-1 score of 0.15 and BLEU-2 score of 0.06 reflect the model's difficulty in achieving lexical overlap between the generated and ground-truth comments. Similarly, the ROUGE scores—ROUGE-1 at 0.23, ROUGE-2 at 0.06, and ROUGE-L at 0.19—indicate limited overlap in n-grams and longest common subsequences between the generated and ground-truth comments. This is particularly evident in the systematic approach of GPT-4's comments, as seen in manual evaluation Example No.1, where the model focused on describing technical details rather than conveying the abstract functionality intended by the developer.

Table IV: Statistics of Arabic Summarization

Statistic	Average	Maximum	Minimum
Word per comment	17.97	49	7

B. RQ2: What is the nature of the Arabic comments generated by these models?

Length: Table IV represents the words per Arabic summarization. To get a better understanding, we compare it with the word counts of English ground-truth comments in Table I. The average word count for Arabic comments is significantly higher at 17.97, compared to just 8.61 for English. This suggests a tendency for GPT-4 to produce more verbose outputs in Arabic, which can be attributed to: **i)** the addition of introductory phrases that are not present in the English ground-truth. For example, as shown in Table V, 44.53% of the comments start with nouns like "تعليق" (comment), which serve as introductory phrases indicating that the text is a comment. These unnecessary phrases contribute to the increased length of the generated Arabic comments compared to the English ground-truth. **ii)** The morphological richness of the Arabic language often requires more words to explain the same concepts as in English. For example, "هذه الوظيفة تقوم بحذف" translates to ("this function deletes.") The English version is shorter because the model added the phrase "تقوم" (does) to the generated comment. The model could have made the comment more concise by rephrasing it as "هذه الوظيفة تحذف" (this function deletes), which omits the unnecessary auxiliary verb "تقوم" and uses the direct verb "تحذف" (deletes). In this case, "تحذف" is a straightforward verb indicating an action, whereas "بحذف" would typically be used in a more complex structure, meaning "by deleting" or "in the act of deleting," and is often part of a longer phrase. **iii)** the model sometimes tends to describe each line of code and its components rather than summarizing the abstract functionality as discussed in RQ1. This behavior contributes to the increased verbosity of the Arabic comments, as it involves detailed descriptions of multiple code elements, which inherently lengthens the comment. The maximum word counts for both languages are relatively similar, with Arabic at 49 and English at 46, indicating that GPT-4 is capable of capturing complex functionalities equally well across both languages. Conversely, the minimum word counts show a high contrast—Arabic comments have a minimum of 7 words, whereas English comments can go as low as 2 words. This highlights a baseline verbosity in Arabic outputs and indicates the model's tendency to avoid overly brief expressions in Arabic despite the clear instructions in the designed prompt to generate "تعليق قصير" (short comment).

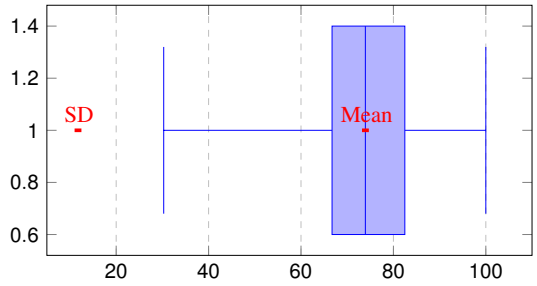

Figure 2: Readability Score of Arabic comments

Part-Of-Speech (POS): Table V shows a significant portion, 44.53%, of the comments begin with a noun. This frequent use of nouns at the start of sentences aligns with common Arabic grammatical structures, where nouns often introduce subjects and objects or are used as an introduction to a sentence. For example, the phrase "تعليق" (comment) establishes the introduction of the sentence, which describes that the text generated is a comment. This pattern suggests that the model often opts for a clear introduction to describe the text.

Furthermore, 29.95% of the comments start with a determiner and noun combination. This structure is frequently used in Arabic to introduce specific elements, providing clarity and context. For instance, the phrase "هذه الدالة" (this function) uses the determiner "هذه" (this) to specify the noun "الدالة" (function), thereby clearly identifying the subject. This syntactic choice helps to ground the comment in a specific context, making it immediately clear the generated comment is related to a function and not to a class or a specific line of code.

Similar to the (DET, NOUN) combination, 29.69% of the comments follow the pattern of determiner, noun, and verb, which indicates an action-oriented style. This structure quickly introduces an action or functionality of the code, as seen in the comment "هذه الوظيفة تقوم" (this function deletes). The inclusion of a verb early in the sentence helps to clarify the action being described.

Table V: Part-of-Speech Analysis of Arabic Comments

Part-of-Speech Pattern	Count	Percentage
First term in a comment Noun (NOUN)	171	**44.53%**
First two terms in a comment Determiner, Noun (DET, NOUN)	115	**29.95%**
First three terms in a comment Determiner, Noun, Verb (DET, NOUN, VERB)	114	**29.69%**

Readability: Figure 2 presents a mean readability score of 74.16, indicating that the majority of the generated comments are generally of ***moderate*** readability for Arabic-speaking developers. The standard deviation of 11.93 reflects a moderate degree of variability in the readability scores, which can be attributed to differences in factors such as code syntax and English vocabulary. For instance, some comments include specialized syntax, like date formatting (e.g., `%Y-%m-%d`), which can affect readability.

The readability scores range from a minimum of 30.29 to a maximum of 100, highlighting the diversity in comment complexity. Some Arabic comments are easy to read, such as "هذه الوظيفة تطبع النسخة الحالية من التطبيق" (This function prints the current version of the application), which lacks complex, lengthy, or Faseeh words and does not contain English terms. On the other hand, comments with a minimum score of readability, e.g., 30.29, often have a higher proportion of English terms, which dominate the Arabic text and contribute to lower readability.

V. CONCLUSION

While the GPT-4 model has proven effective in advancing various software engineering tasks, including code summarization, its performance in generating Arabic code comments reveals significant limitations. Although GPT-4 can produce correct Arabic summaries, these comments often fail to align with the developer's intent, as evidenced by the manual analysis and low ROUGE and BLEU scores. The moderate BERT-Similarity score indicates that the model captures some functional aspects of the code but frequently misses the intended meaning. Moreover, the readability of the generated comments varies, with some being easy to understand and others falling short. These findings highlight the need for further refinement of LLMs like GPT-4 to support non-English languages in the context of code documentation. In the future, we plan to benchmark multiple LLMs for the Arabic code summarization.

REFERENCES

[1] W. U. Ahmad, S. Chakraborty, B. Ray, and K.-W. Chang. A transformer-based approach for source code summarization. *arXiv preprint arXiv:2005.00653*, 2020.

[2] T. Ahmed and P. Devanbu. Few-shot training llms for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, pages 1–5, 2022.

[3] T. Ahmed and P. Devanbu. Learning code summarization from a small and local dataset. *arXiv preprint arXiv:2206.00804*, 2022.

[4] T. Ahmed, K. S. Pai, P. Devanbu, and E. Barr. Automatic semantic augmentation of language model prompts (for code summarization). In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.

[5] A. Aljohani and H. Do. From fine-tuning to output: An empirical investigation of test smells in transformer-based test code generation. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 1282–1291, 2024.

[6] C. E. A. Coello, M. N. Alimam, and R. Kouatly. Effectiveness of chatgpt in coding: a comparative analysis of popular large language models. *Digital*, 4(1):114–125, 2024.

[7] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira. A study of the documentation essential to software maintenance. In *Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information*, pages 68–75, 2005.

[8] M. El-Haj and P. Rayson. Osman—a novel arabic readability metric. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 250–255, 2016.

[9] 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.

[10] J. L. Fleiss, B. Levin, M. C. Paik, et al. The measurement of interrater agreement. *Statistical methods for rates and proportions*, 2(212-236):22–23, 1981.

[11] X. Hu, G. Li, X. Xia, D. Lo, and Z. Jin. Deep code comment generation. In *Proceedings of the 26th conference on program comprehension*, pages 200–210, 2018.

[12] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

[13] M. T. I. Khondaker, A. Waheed, E. M. B. Nagoudi, and M. Abdul-Mageed. Gptaraeval: A comprehensive evaluation of chatgpt on arabic nlp. *arXiv preprint arXiv:2305.14976*, 2023.

[14] J. Liu, C. S. Xia, Y. Wang, and L. Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023.

[15] Y. Liu, T. Le-Cong, R. Widyasari, C. Tantithamthavorn, L. Li, X.-B. D. Le, and D. Lo. Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Transactions on Software Engineering and Methodology*, 33(5):1–26, 2024.

[16] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation (2021). *arXiv preprint arXiv:2102.04664*.

[17] P. W. McBurney and C. McMillan. Automatic source code summarization of context for java methods. *IEEE Transactions on Software Engineering*, 42(2):103–119, 2015.

[18] C. Piech and S. Abu-El-Haija. Human languages in source code: Auto-translation for localized instruction. In *Proceedings of the Seventh ACM Conference on Learning@ Scale*, pages 167–174, 2020.

[19] P. Rani, M. Birrer, S. Panichella, M. Ghafari, and O. Nierstrasz. What do developers discuss about code comments? In *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 153–164. IEEE, 2021.

[20] D. Roy, S. Fakhoury, and V. Arnaoudova. Reassessing automatic evaluation metrics for code summarization tasks. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1105–1116, 2021.

[21] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[22] F. A. Sakib, S. H. Khan, and A. Karim. Extending the frontier of chatgpt: Code generation and debugging. *arXiv preprint arXiv:2307.08260*, 2023.

[23] E. Shi, Y. Wang, L. Du, J. Chen, S. Han, H. Zhang, D. Zhang, and H. Sun. On the evaluation of neural code summarization. In *Proceedings of the 44th international conference on software engineering*, pages 1597–1608, 2022.

[24] W. Sun, C. Fang, Y. You, Y. Miao, Y. Liu, Y. Li, G. Deng, S. Huang, Y. Chen, Q. Zhang, et al. Automatic code summarization via chatgpt: How far are we? *arXiv preprint arXiv:2305.12865*, 2023.

[25] M. Tawkat Islam Khondaker, A. Waheed, E. Moatez Billah Nagoudi, and M. Abdul-Mageed. Gptaraeval: A comprehensive evaluation of chatgpt on arabic nlp. *arXiv e-prints*, pages arXiv–2305, 2023.

[26] L. Wang, N. Yang, X. Huang, L. Yang, R. Majumder, and F. Wei. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[27] Y. Wang, W. Wang, S. Joty, and S. C. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*, 2021.

[28] S. N. Woodfield, H. E. Dunsmore, and V. Y. Shen. The effect of modularization and comments on program comprehension. In *Proceedings of the 5th international conference on Software engineering*, pages 215–223, 1981.

[29] C. Zhang, J. Wang, Q. Zhou, T. Xu, K. Tang, H. Gui, and F. Liu. A survey of automatic source code summarization. *Symmetry*, 14(3):471, 2022.