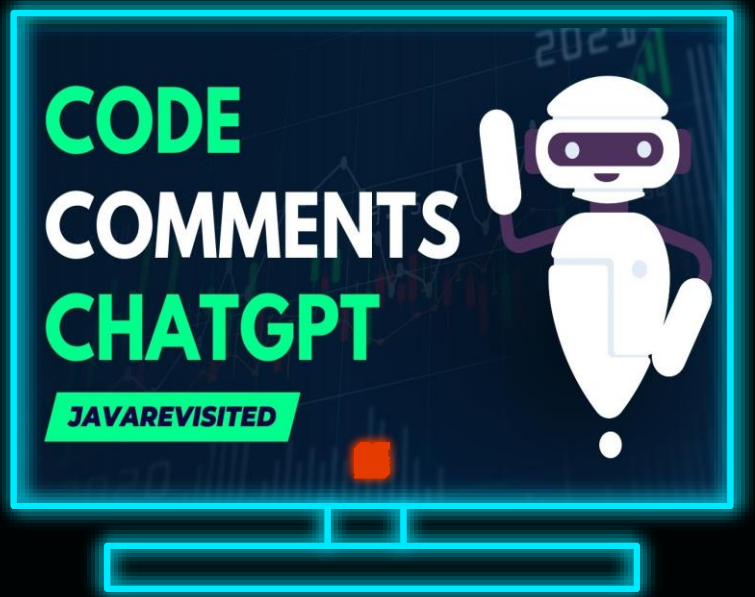


ChatGPT's Ability to generate comments for codes

Hassan
Abdul-Mohsen
Abdullah
Mishary

Introduction

- Exploring LLMs abilities and Limitations.
- How well can it write code comments?
- Is Human-Documentation Superior

01

“A Comparative Analysis of Large Language Models for Code Documentation Generation”

Idea

- Evaluate the effectiveness of LLMs in generating code documentation
- comparing Ai models across different documentation levels (inline, function, and file-level).

AI Models Used:

01 GPT-3.5, GPT-4

02 Bard

03 Llama2

04 StarChat - Beta

Method

- ❑ checklist-based evaluation approach
- ❑ 14 Python code snippets from well-documented open-source projects
  (Bitcoin, TensorFlow, SQLAlchemy, Scikit-Learn, and Reddit)
- ❑ generate documentation using prompts
- ❑ two expert reviewers assess the output

Evaluation metrics used:

01 Accuracy

02 Completeness

03 Relevance

04 Understandability

05 Readability

06 Time Taken

Results

#1

ChatGPT

Best quality, time consuming

#2

Bard

Superior to Human-Documentation

#3

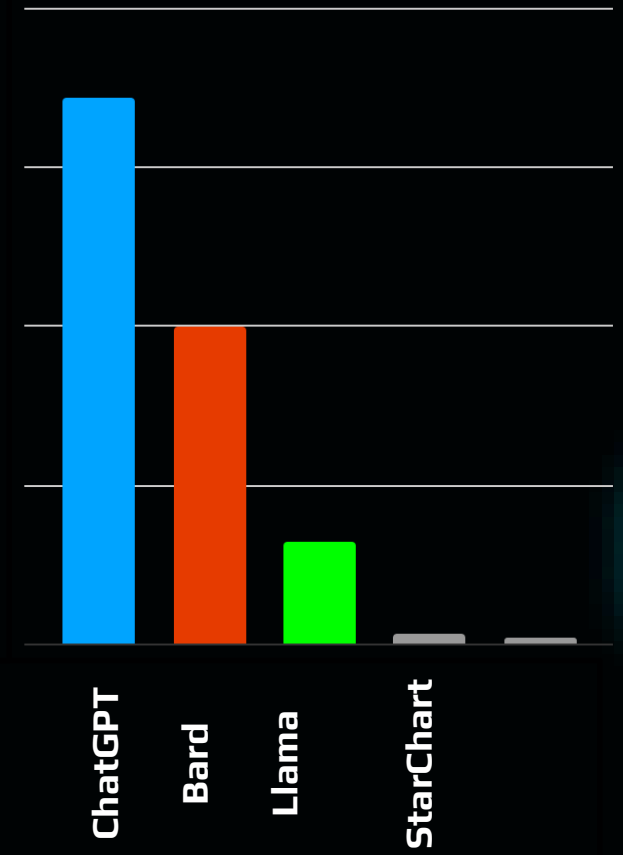
Llama

Performance Lacking

#4

StarChat

Weakest Comment Generation



02

“Automatic Code Summarization via ChatGPT: How Far Are We?.”

Idea

- investigate ChatGPT's effectiveness in automatic code summarization
- comparing performance against state-of-the-art (SOTA) models

Method

- ❑ Design prompts to guide ChatGPT in generating comments for code snippets
- ❑ Utilize the CSN-Python dataset for code test set.
- ❑ Compare summaries with those produced by SOTA models

Evaluation Metrics

- measure the quality of generated text against reference summaries, using the following metrics:

- BLEU
- METEOR
- ROUGE-L

Table 5: Performance of ChatGPT with different prompts

Method	CSN-Python		
	BLEU	METEOR	ROUGE-L
ChatGPT (short)	4.85	12.90	14.41
ChatGPT (one sentence)	10.66	13.36	20.37
ChatGPT (within 40 words)	4.83	13.01	14.05

Table 6: Performance of ChatGPT and three SOTA models

Method	CSN-Python		
	BLEU	METEOR	ROUGE-L
NCS	15.8	10.6	31.3
CodeBERT	18.7	12.4	34.8
CodeT5	20.0	14.7	37.7
ChatGPT (one sentence)	10.28	14.40	20.81

Results

- ChatGPT's performance was significantly lower than SOTA models.
- ChatGPT's code summaries were less aligned with reference summaries than those produced by other models

03

“Using GPT-4 for Source Code Documentation.”

Idea

- Investigates GPT-4's effectiveness
- generating Java class-level and method-level documentation
- compare to manually written reference documentation.

Method

- using predefined prompts.
- GPT-4 generates Javadoc comments for class and method levels.
- Output is evaluated using both automated (BLEU, ROUGE) and manual methods.
- Select Eleven Java classes and 55 methods from open-source projects

Evaluation metrics used:

01 Correctness

02 Completeness

03 Conciseness

04 BLEU

05 ROUGE-1

Results

- **Document Level:** GPT-4 is more accurate than the reference but slightly less complete.
- **Class Level:** GPT-4 performed poorly in completeness, correctness was high.
- **Method-level:** BLEU and ROUGE scores were significantly higher than other levels.

04

“Can Developers Prompt? A Controlled Experiment for Code Documentation Generation.”

Idea

- Study how well both students and professionals, can prompt LLMs to generate code documentation
- compare ad-hoc prompts with predefined few-shot prompts.

Method

- controlled, between-subject experiment
- Used 2 Python functions to document
- Ad-hoc group craft unique prompts
- Control group used pre-defined prompts

Evaluation metrics used:

01 Correctness

02 Missing Information

03 Unnecessary Information

04 Usefulness

05 Helpfulness

06 Readability

Results

- Predefined prompts produced higher-quality documentation.
- Professionals performed better with ad-hoc than students
- Both tools improved code understanding.
- Tools performed best with complex functions

05

“The Role of ChatGPT in Computer programing.”

Idea

- explore the role and capabilities of ChatGPT in computer programming
- focus on how ChatGPT can assist developers
- Study ChatGPT ability to offer features such as code completion, error correction, optimization, and documentation generation

Method

- Giving ChatGPT prompts.

- Analyzing its ability to:

1) Predict code snippets
2) Fix syntax errors
3) Suggest optimization
4) Generate documentation

Evaluation Metrics

- No formal evaluation metrics are mentioned in the research.
- the effectiveness of ChatGPT is assessed qualitatively through examples
- Accuracy in code suggestions, error detection, and document generation.

Results

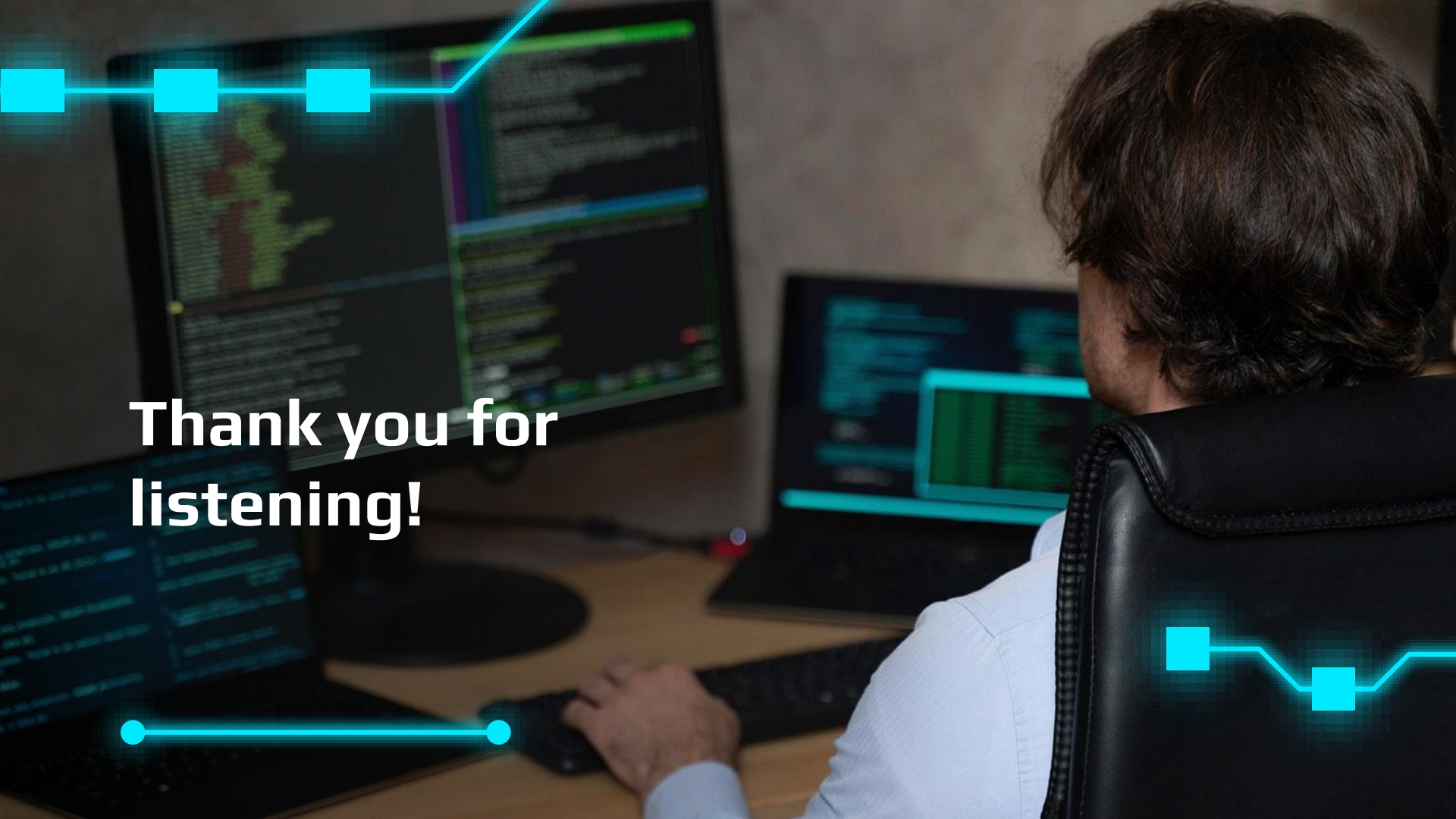
- ChatGPT is a valuable tool for developers, helps reduce coding errors, improve code quality, and accelerate development processes.
- ChatGPT is effective in providing automating documentation and refine code structure.
- ChatGPT still requires human supervision for tasks.

06

CONCLUSION

LLMs showed better results than human-documentation

LLMs are reliable in code documentation
ChatGPT showed better performance than other LLMs


Thank you for
listening!