# Evaluating Code Understanding in APIs

Abhishek Som
University of Illinois Urbana-Champaign
USA
asom2@illinois.edu

Neel Harip
University of Illinois Urbana-Champaign
USA
nhari7@illinois.edu

## ABSTRACT

Recent advancements in AI have led to the development of code understanding APIs that assist developers with writing, documenting, and debugging code. This tech review focuses on evaluating three major APIs: OpenAI Codex, Amazon CodeWhisperer, and Google Codey. Our objective is to assess their effectiveness in three key tasks: code summarization, documentation generation, and bug detection across Python and C++. We will define evaluation metrics to benchmark the APIs and provide a comparative analysis highlighting their strengths and limitations, offering practical insights for developers and researchers.

## 1 INTRODUCTION

The field of software engineering has seen a significant transformation with the rise of AI-driven code understanding APIs. These tools promise to enhance developer productivity by offering features such as code summarization, automatic documentation generation, and bug detection. Evaluating the effectiveness of these APIs is essential to understanding their practical utility and guiding their adoption in software development workflows.

## 2 PROPOSED WORK

This tech review focuses on evaluating three major code understanding APIs: OpenAI Codex, Amazon CodeWhisperer, and Google Codey. Our evaluation will target three primary tasks across Python and C++:

- Code Summarization: Assessing the APIs' ability to generate accurate and concise summaries of given code snippets.
- Documentation Generation: Evaluating the clarity and usefulness of auto-generated code documentation.
- Bug Detection: Analyzing the APIs' ability to identify and suggest fixes for code issues.

## 3 METHODOLOGY

Our approach will involve the following steps:

(1) Select standardized code snippets in Python and C++ that cover diverse use cases.
(2) Run these snippets through each API and collect their outputs.
(3) Evaluate the outputs using predefined metrics: accuracy, clarity, and usefulness.
(4) Conduct a comparative analysis to identify the strengths and limitations of each API.

## 4 EVALUATION METRICS

The evaluation will be based on the following metrics:

- **Accuracy**: How correctly the API summarizes or explains a piece of code.
- **Clarity**: How understandable and concise the generated documentation is.
- **Usefulness**: The relevance of the suggested bug fixes and code improvements.

## 5 COMPLEXITY ASSESSMENT

This project involves defining robust evaluation criteria, crafting diverse test cases for Python and C++, and ensuring unbiased assessments of the APIs. Substantial analytical effort is required to derive meaningful insights from the API outputs. Additionally, detailed documentation and reporting will be essential to capture the findings comprehensively.

## 6 EXPECTED OUTCOMES

This tech review aims to provide a clear comparison of the strengths and weaknesses of each API. We expect to identify:

- Which API performs best for specific programming tasks.
- The limitations in handling complex code structures.
- Recommendations for developers on the best API for different use cases.