LLM of Babel: Evaluation of LLMs on code for non-English use-cases

Yongcheng Huang
EEMCS, Delft University of Technology, The Netherlands

Supervisors: Prof. Dr. Arie van Deursen , Assistant Prof. Dr. Maliheh Izadi , ir. Jonathan Katzy

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Yongcheng Huang
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Arie van Deursen , Assistant Prof. Dr. Maliheh Izadi, Assistant Prof. Dr. Gosia Migut

LLM of Babel: Evaluation of LLMs on code for non-English use-cases

Yongcheng Huang
Delft University of Technology
Delft, The Netherlands

Abstract

After the emergence of BERT, Large Language Models (LLMs) have demonstrated remarkable multilingual capabilities and have seen widespread adoption globally, particularly in the field of programming. However, current evaluations and benchmarks of LLMs on code primarily focus on English use cases. In this study, we assess the performance of LLMs in generating Chinese Java code comments through open coding. Our experiments highlight the prevalence of model-specific and semantic errors in generating Chinese code comments using LLMs, while also revealing a relative absence of grammatical issues due to the unique characteristics of the Chinese language. Additionally, we validated the potential for quantitatively analyzing semantic errors, especially Hallucinations, by examining the cosine similarity of word embeddings. Our findings propose an Error Taxonomy for evaluating LLMs on code in non-English scenarios and demonstrate the possibilities of using cosine similarity of word embeddings to judge the quality of code comment generation.

Keywords

Large Language Model, Multilingual Code, Natural Language Processing, Code Inference, Non-English Programming, Code Comments, Error Taxonomy, Model Evaluation, AI in Software Development, Computational Linguistics, Cosine Similarity

1 Introduction

LLMs are increasingly showing their vast capabilities in multiple areas in recent years, driven by advancements in universal models and the proliferation of user-friendly online platforms offered by various providers, including OpenAI ChatGPT series. The integration of LLMs into programming has proven beneficial across several domains. Research highlights include enhanced productivity among programmers [1], an improvement in the knowledge base across all levels of expertise [2], and the facilitation of programming education for novices [3]. These developments reflect a substantial shift in how programming skills are acquired and applied in the industry, as evidenced by their use in major corporations such as Alibaba Group [4]. This underscores the crucial impact of LLMs in contemporary software development.

Despite the promising capabilities of LLMs, significant performance discrepancies exist when these models are utilized with different natural languages [5]. A study conducted by Microsoft highlights a significant gap [6], with approximately 1.2 billion people and over 6000 languages underserved by current LLM technologies. Even among widely spoken languages, discrepancies in model performance persist, underscoring an uneven technological reach. Recent scholarly work, including studies on the multilingual capabilities of LLMs [7] and explorations of the safety challenges presented in multilingual contexts [8], indicating growing recognition of these issues. The majority of available LLMs are trained on datasets from online repositories such as GitHub, which are cost-effective and easily accessible. However, this data is predominantly in English, and the models are evaluated using English-based code benchmarks such as CodeXGlue [9] and HumanEvalX [10].

While models like BERT [11] has demonstrated their efficacy in multilingual tasks, the focus on non-English language performance, particularly in programming contexts, remains inadequately addressed. This presents a major issue: Although programming with LLMs is very common in non-English environments, especially Chinese, we do not have a systematic analysis of its performance. Consequently, the performance of these models in generating Chinese comments remains highly uncertain and potentially inadequate [12]. Since the gap in research and application highlights a critical area for further investigation and development, we derive our first research question.

In this study, our first research question is: **What mistakes do LLMs make when generating Chinese Java code comments?** To answer this question, we implement a detailed analysis and annotation of inference outcomes to construct an error taxonomy. This enables us to implement a comprehensive quantitative and qualitative analysis to the inference result from `StabilityAI/stable-code-3b` [13].

Furthermore, we aim to address the second research question in this paper: **Can semantic errors be analyzed through the examination of cosine similarities of word embeddings?** By applying cosine similarity [14], we want to identify semantic discrepancies at the level of word embeddings. Our approach involves analyzing the similarity between the original comment and the reference comment, and comparing it with the similarity between the original comment and the comment containing semantic errors.

This study reveals that large language models have a large number of semantic and model-specific problems in the problem of Chinese Java code comment generation. However, due to the particularity of the Chinese language, the grammatical problems that were frequently found in our other studies are not common. At the same time, we verified the possibility of using cosine similarity to compare word embeddings to detect semantic errors by comparing cosine similarity.

Through this research, we hope to build the Babel Tower for LLMs in different language code comment generation scenarios. We also hope to extend our research results to other non-English languages and establish a multilingual evaluation standard in the `LLM4Code` field. At the same time, since we have verified the feasibility of using cosine similarity to detect semantic errors, automated analysis has also become possible through this fact.

2 Related Work

The use of LLMs is a prominent topic in deep learning, and numerous studies have explored the potential errors that can arise when deploying these models. This problem is even more serious in the application scenario of LLMs in non-English languages. Widely used metrics such as BLEU [15] and ROUGE [16] have significant deficiencies in this regard. These metrics, focused on n-gram overlap, are limited in capturing fine-grained grammatical and semantic errors in code generation.

In previous research on evaluating the quality of code comments, the innovative combination of machine translation metrics and open coding has yielded impressive results [17]. This approach has proven particularly effective in handling the intricacies of semantic evaluation, demonstrating its potential in addressing complex linguistic tasks. The success of these methodologies has provided valuable insights and a strong foundation for further exploration in the field. It has particularly inspired our current research to delve deeper into understanding the specific types of errors LLMs make, especially in the context of generating Chinese Java code comments. This evaluation aims to build on existing knowledge while addressing the unique challenges presented by Chinese applications.

To understand why these errors occur, we can look at research on LLMs that highlights limitations associated with standard likelihood maximization in Natural Language Generation (NLG) [18]. Early studies like Grid Beam Search (GBS) [19] and neural text generation [20] noted that maximizing likelihood can lead to degeneration, described as hallucination [21]. Moreover, LLMs struggle with semantic accuracy, requiring significant manual work to develop rule-based NLP systems or training corpora. Semantic spaces [22] were proposed to capture meaning in natural language representations, but semantic issues remain prevalent.

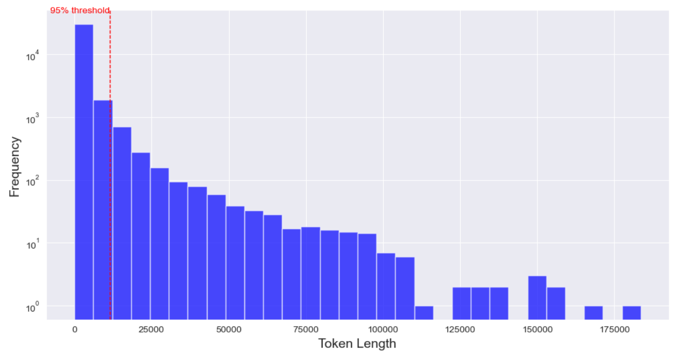
Epistemic and aleatoric uncertainties have been used to assess the reliability of LLM outputs, demonstrating the need for skepticism towards LLM-generated content [23]. While many optimizations have been made for English, there's a lack of systematic analysis for Chinese generation, particularly in tasks like Chinese inference for Java code comments.

Due to the reliance of large language models on word embeddings , Cosine similarity, a common metric in information retrieval, has been effective in studying semantics and generation quality [14] [24] [25]. Recent studies have shown improvements in quantifying semantic similarity using cosine similarity in deep learning [26].

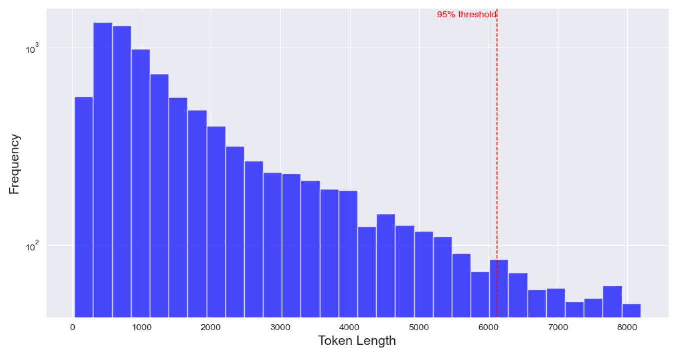
In a recent study, SemScore, which utilizes semantic textual similarity (STS) to evaluate the quality of model outputs, demonstrated the feasibility of assessing large models based on semantic similarity [27]. This approach provides a promising alternative for rapidly determining the desirability of model outputs. The insights gained from SemScore have inspired our second research question, guiding us to explore the application of semantic similarity metrics in the evaluation of LLM-generated content with semantic errors.

3 Methodology

In evaluating the performance of large language models (LLMs) in multilingual programming environments, we employ several key approaches: Data Preprocessing, Inference Pipeline, Open Coding, Cosine Similarity and Principle Component Analysis, Visual Data Annotation, and Ernie 1.0. These methodologies allow us to thoroughly evaluate the capabilities of LLMs and identify specific challenges in generating accurate and semantically meaningful outputs across different languages.



(a) Distribution of the file lengths in the initial dataset.



(b) Distribution of the file lengths in the new dataset.

Figure 1: Distribution of file lengths.

3.1 Data Preparation

For this research, we created a Chinese Java code dataset by searching for the 2500 most frequent Chinese words on GitHub, based on the study "Most Common Words by Language" [28]. We used Regular Expression (REGEX) to identify and categorize comments within the retrieved code files. To ensure consistency and reliability across models and datasets, we set the Maximum File Length (Token) and Word Inclusion Parameters as hyperparameters. Given that the codegemma-7b model [29] supports up to 8192 tokens, we set this as the upper limit for file length. To ensure the generated comments are in the same language as the original, we retain two characters for Chinese in line comments and three characters for block comments.

We preprocessed our dataset based on file length, content, and language [30] and made it available as open-source on Hugging Face [31]. For this research, we used the 95th percentile as the threshold for maximum comment length, accounting for frequent outliers. Figure 1 shows the distribution of file lengths in both datasets. The original dataset consists of 34,300 files, while the new dataset contains 9,285 files. The red dashed lines in the tables indicate the 95th percentile of comment length, which is 199 tokens in the processed dataset.

3.2 Inference Pipeline

To answer our first research question, we designed an inference pipeline for handling datasets from Hugging Face, outputting results compatible with our data visualizer outlined in Section 3.5.

The pipeline processes masked data by replacing Fill In Middle (FIM) placeholders with the `<fim_middle>` tag used during the training of Stable-code-3b. Using FIM models allows us to generate higher quality, more accurate, and contextually relevant comments [32], avoiding biases and limitations of prompt-based models. After processing, the data is input into the model for inference, with the output automatically uploaded to HF for future work and open-source sharing. Additionally, we have implemented a universal pipeline to evaluate datasets in other languages, minimizing non-human errors and maintaining consistency.

3.3 Open Coding

This project aims to bridge the language gap in non-English programming environments by analyzing the performance of LLMs. To achieve this, we have adopted a robust qualitative analysis method known as Open Coding [33], integral to Grounded Theory [34], to dissect and understand the subtleties of multilingual LLM output. Open Coding involves breaking down data into discrete components that can be examined and compared. In our project, this translates to analyzing code comments and the corresponding outputs from LLMs when tasked with non-English commenting challenges. Each piece of comment, along with its output, is meticulously examined to identify distinct errors such as Linguistic Errors, Semantic-specific nuances, and Model-specific Problems in output. These are labeled with descriptive names that encapsulate their essence, thus building a library of identifiable phenomena that are critical for our analysis.

Open Coding allows us to create specific labels and categories that make the qualitative data comprehensible at a granular level. For instance, if an LLM generates incorrect syntax or language constructs due to language differences, these are tagged and categorized under specific error types like "Model-specific, Too specific, M-TS" or "Linguistic, Usage of incorrect synonym, L-IS." Through the iterative process of Open Coding, we develop a taxonomy of common failures and limitations observed in LLMs when dealing with non-English code. This taxonomy is pivotal in understanding the breadth and depth of challenges faced by LLMs across different languages, thus directing future improvements.

Our research team comprises five researchers working across four different languages. In each round, we manually label 200 comments produced by different models in a specific language. This process is repeated for three rounds, resulting in a comprehensive examination of the LLM outputs. From the errors identified in different languages, we derive an error taxonomy, which has been refined through 11 versions. This systematic approach ensures a thorough and nuanced understanding of the challenges and capabilities of LLMs in non-English programming contexts. In this way, we can clearly determine the types of errors generated by large models and generate error taxonomy to evaluate the performance of other large models.

3.4 Cosine Similarity and Principal component analysis

Cosine similarity is crucial in data analysis for assessing the similarity between two non-zero vectors by computing the cosine of the angle between them. This metric captures the orientation of vectors, making it ideal for comparing data embeddings based on directionality alone. It is advantageous in natural language processing (NLP) and machine learning, where text or items are converted into high-dimensional embeddings that encapsulate semantic features.

$$Cosine(x, y) = \frac{x \cdot y}{|x||y|}$$

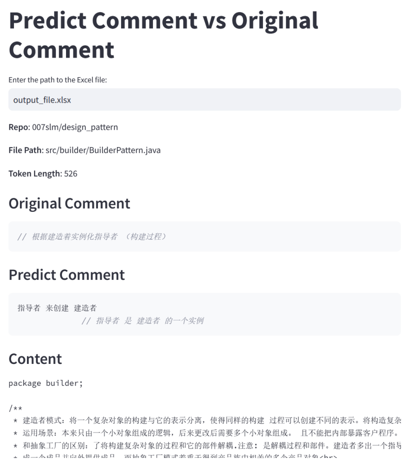
In this experiment, we will use cosine similarity to analyze "Too General" (SE-TG) and "Hallucination" (SE-HA) errors. By comparing the cosine similarity between overly broad and specific reference annotations, we want to find the relationship between semantics and cosine similarity. The dataset comprises 50 sets of comments divided into line and block types, each with standard and "too general" versions. To ensure the reliability of our semantic similarity metrics, control groups were established. In the line comment analysis, one control group used a single-sentence Chinese comment, `"This is a function that calculates addition"`, and compared it against its reverse string. This juxtaposition aimed to demonstrate how semantic structures are disrupted when the sequence of the characters is inverted. The second control group employed the original sentence as both a normal and a broad comment to establish a baseline, ensuring that identical items achieved a cosine similarity score of 1. This approach helps validate the reliability of our similarity metrics by confirming that identical texts are recognized as such, achieving perfect similarity.

This study also investigates the use of cosine similarities to detect two types of hallucinations—Out of Context Hallucinations and Educated Guesses—using a dataset of 52 hallucinated examples. We compared cosine similarities between a reference code comment and both a normal and a hallucinated comment, labeled Group1 and Group2, respectively. The hallucinated comments were generated by stable-code-3b. Additionally, Principal Component Analysis (PCA) was employed on word embeddings to further validate the correlation between cosine similarity and semantic discrepancies.

3.5 Visual data annotation

For this research, we developed a data visualization tool using Streamlit [35] to enhance the analysis and validation of outputs from LLMs as visualized in Figure 2. This application processes output data encapsulated in an Excel file, which contains both the original comments and the model-generated inferences, enabling real-time, interactive evaluation and categorization of errors directly within the visualization interface.

Functionality within the Streamlit app includes a sidebar-controlled pagination system allowing detailed, entry-by-entry review. For each data entry, the app displays metadata such as the repository name, file path, and the length of tokens, alongside the content of both the original and predicted comments. This setup facilitates



Predict Comment vs Original Comment

Enter the path to the Excel file:

output_file.xlsx

Repo: 007slm/design_pattern

File Path: src/builder/BuilderPattern.java

Token Length: 526

Original Comment

```
// 根据建造者实例化指导者 （构建过程）
```

Predict Comment

```
指导者 来创建 建造者
            // 指导者 是 建造者 的一个实例
```

Content

```
package builder;

/**
 * 建造者模式：将一个复杂对象的构建与它的表示分离，使得同样的构建 过程可以创建不同的表示。将构造复杂
 * 运用场景：本来只由一个小对象组成的逻辑，后来更改后需要多个小对象组成。 且不能把内部暴露客户程序。
 * 和抽象工厂的区别：了将构建复杂对象的过程和它的部件解耦.注意：是解耦过程和部件。建造者多出一个指导
 * 成一个成品并向外提供成品。而抽象工厂模式着重于得到产品族中相关的多个产品对象<br>
```

Figure 2: Data Visualizer Example.

an immediate and clear comparison between expected and actual model outputs, essential for assessing the model's performance.

3.6 Ernie 1.0

During the research of the first research question, we utilized the stable-code-3b model, which is a decoder-only system. For the study for research question 2, it is essential to employ a model with encoder capabilities, specifically an open-source encoder, to enable the translation from Chinese inference results to embeddings. The stable-code-3b model lacks this feature. Therefore, we have decided to transition to Ernie 1.0 [36], which supports both encoding and decoding functionalities and is explicitly trained on Chinese datasets. This model not only meets our requirements for an open-source encoder but also provides robust support for Chinese language inference and tokenization, making it ideal for our needs.

4 Results

In this section, we present our results in the following two areas: : a comprehensive qualitative analysis using an error taxonomy, and a quantitative analysis based on cosine similarity measurements.

4.1 Qualitative Analysis

As a result of the open coding process, we developed a Hierarchical Error Taxonomy, illustrated in Table 1. This taxonomy categorizes the various errors identified during our analysis and includes the counts for each error type: Linguistic Error, Semantic Error, and Model Specific Error. Each of these categories is further subdivided to capture more specific error types. For instance, Linguistic Errors include issues like Grammar, Incorrect Synonyms, and Wrong Language Usage, while Semantic Errors encompass Incomplete Descriptions and Overly General Statements. Model Specific Errors range from Memorization to Random Words and even No Generation, reflecting the unique challenges encountered with specific model behaviors.

To illustrate the error distribution more clearly, Figure 3 is a bar chart showing the error frequency of the labeling result. From a map of where our model, stable-code-3b, fails in Chinese code

Table 1: Taxonomy of failure categories

Failure category plus label ID	Count
Model Specific Errors	**134**
└ (MS-RW) Random Words	5
└ (MS-CC) Copy Context	23
└ (MS-ME) Memorization Level	27
└ (MS-ME1) Personally identifiable information (PII)	8
└ (MS-ME2) Contains URL	4
└ (MS-ME3) Training set memorization	15
└ (MS-ES) Early Stop	4
└ (MS-LS) Late Stop	24
└ (MS-NG) No generation	31
└ (MS-RE) Repetition	16
└ (MS-RE1) Pattern repetition	14
└ (MS-RE2) Verbatim repetition	2
└ (MS-TS) Too Specific	4
Linguistic Errors	**0**
└ (LG-GR) Grammar	0
└ (LG-GR1) Plurality	0
└ (LG-GR2) Conjugation	0
└ (LG-GR3) Gendering	0
└ (LG-GR4) Language Syntax	0
└ (LG-GR5) Capitalization	0
└ (LG-GR6) Cohesion	0
└ (LG-IS) Usage of incorrect synonym	0
└ (LG-WL) Wrong language	0
└ (LG-WL1) Undesired Translations	0
└ (LG-WL2) Incorrect Language	0
Semantic Errors	**197**
└ (SE-TG) Missing Details	21
└ (SE-HA) Hallucination	113
└ (SE-HA1) Misplaced Facts	9
└ (SE-HA2) Out of Context Hallucination	74
└ (SE-HA3) Educated Guess	30
└ (SE-CS) Completion includes code snippet	63
└ (SE-CS1) Commented out code	5
└ (SE-CS2) Code intended to run	58
Syntax Errors	**27**
└ (ST-IF) Incorrect comment format	27
└ (ST-IF1) Comment Style Inconsistency	10
└ (ST-IF2) Omitted Identifier	17
(E) Excluded	**14**
(M) Misc	**7**

commenting tasks, we can find a high incidence of semantic errors,

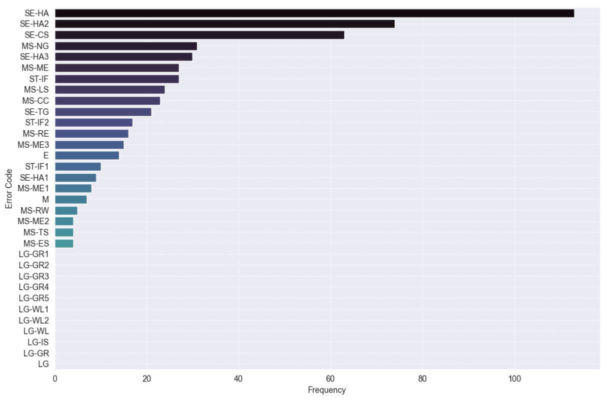
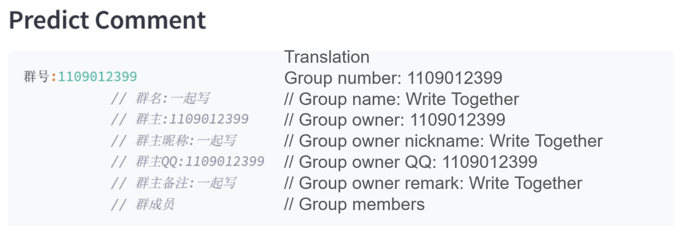


Figure 3: Error frequency of the labeling result.

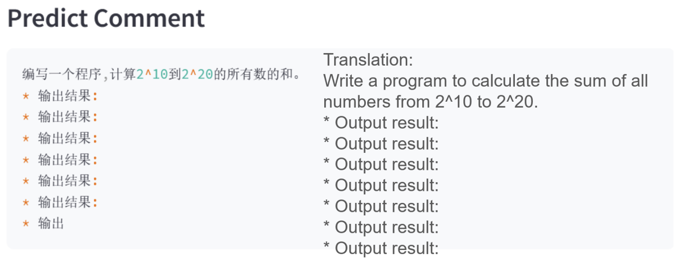


Predict Comment

```
群号:1109012399
            // 群名:一起写
            // 群主:1109012399
            // 群主昵称:一起写
            // 群主QQ:1109012399
            // 群主备注:一起写
            // 群成员
```

Translation
Group number: 1109012399
// Group name: Write Together
// Group owner: 1109012399
// Group owner nickname: Write Together
// Group owner QQ: 1109012399
// Group owner remark: Write Together
// Group members

(a) Error MS-ME1 example (Personally identifiable information included in the generated comment)

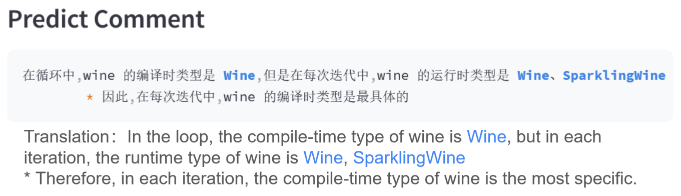


Predict Comment

```
编写一个程序,计算2^10到2^20的所有数的和。
* 输出结果:
* 输出结果:
* 输出结果:
* 输出结果:
* 输出结果:
* 输出结果:
* 输出
```

Translation:
Write a program to calculate the sum of all numbers from 2^10 to 2^20.
* Output result:
* Output result:
* Output result:
* Output result:
* Output result:
* Output result:
* Output result:

(b) Error MS-RE example (Model generates and repeats what it has already said it in some capacity).

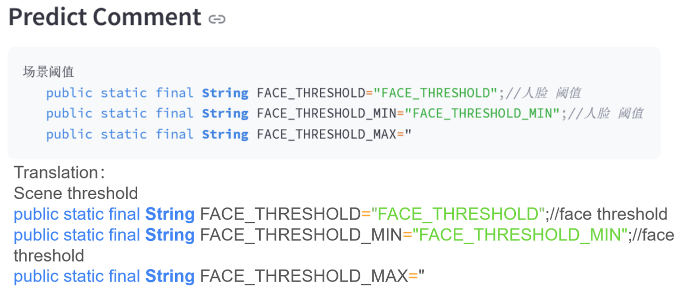


Predict Comment

```
在循环中,wine 的编译时类型是 Wine,但是在每次迭代中,wine 的运行时类型是 Wine、SparklingWine
         * 因此,在每次迭代中,wine 的编译时类型是最具体的
```

Translation: In the loop, the compile-time type of wine is Wine, but in each iteration, the runtime type of wine is Wine, SparklingWine
* Therefore, in each iteration, the compile-time type of wine is the most specific.

(c) Error SE-HA2 example (Out of Context Hallucination).



Predict Comment

```
场景阈值
    public static final String FACE_THRESHOLD="FACE_THRESHOLD";//人脸 阈值
    public static final String FACE_THRESHOLD_MIN="FACE_THRESHOLD_MIN";//人脸 阈值
    public static final String FACE_THRESHOLD_MAX="
```

Translation:
Scene threshold
public static final String FACE_THRESHOLD="FACE_THRESHOLD";//face threshold
public static final String FACE_THRESHOLD_MIN="FACE_THRESHOLD_MIN";//face threshold
public static final String FACE_THRESHOLD_MAX="

(d) Error SE-CS example (Model generates actual code outside of comment).

Figure 4: Examples of common errors.

particularly hallucinations. Among the 300 comments labeled in the final round of open coding, we marked a total of 379 labels, and the Semantic Error (SE) class accounted for 197 of them. Model-specific errors are the second most serious problem, accounting for 134 errors, mainly concentrated in Memorization (MS-ME) and No Generation (MS-NG) problems. Syntax error, which represents the problem of generating incorrectly formatted comments, only accounts for about 7% of the error labels, indicating that this problem is not prominent in the Chinese inference result.

However, there is no Grammar Error labeled during the analysis process, which is due to the peculiarity of Chinese language. For this study, we specifically investigate comments in the Chinese language, which exhibits a number of challenges unique to this linguistic context.

The categories of Excluded (E) and Miscellaneous (M) errors represent less prevalent errors. These labels either involve improper and unrelated comments with code or contain errors unrelated to the model's inference. All unrelated or unnecessary data will not be included in this research to ensure that the study focuses on the core errors of the large language model during inference.

For those common errors identified during the manual review of comment inferences, we will provide some examples to illustrate them more clearly. All comments are originally in Chinese and there is an English translation added only for reading. Figure 4a illustrates a case where the model inadvertently included personally identifiable information in the generated comment, which did not exist in the original prompt, raising concerns about privacy and data integrity. In Figure 4b, the model exhibits an issue of redundancy by regenerating content that has already been produced, which can lead to inefficiencies and decreased user trust. Figure 4c showcases a comment where the model introduces hallucinated information about a hypothetical class named 'wine', which was not present in the original input, highlighting the model's tendency to fabricate irrelevant details. To ensure a rigorous standard for categorizing hallucination problems, we refer to the investigation [25]on hallucination detection and mitigation, where the author provides valuable insights into the various categories of hallucinations. Lastly, Figure 4d presents an example where the model generates a code snippet outside of the expected comment context, illustrating its capability to produce complex outputs but also the potential for generating out-of-context information.

4.2 Quantitative Analysis

In our previous research, we identified that deriving an Error Taxonomy through manual labeling is prohibitively expensive. Our qualitative analysis revealed a higher error rate in semantic aspects of the model. Consequently, we want to focus further on semantic

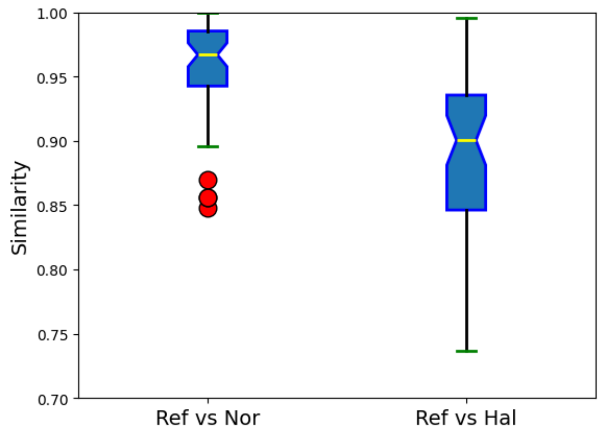


Figure 5: Cosine similarities between two groups of embeddings.

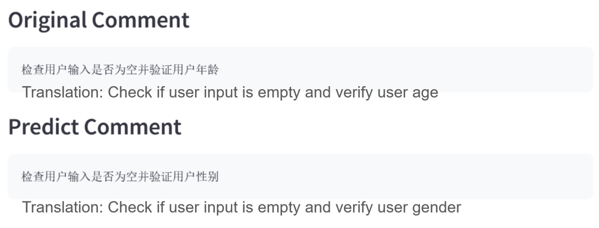
errors, bridging the gap between qualitative and quantitative analysis by exploring the relationship between semantic errors and the cosine similarity of comment word embeddings.

4.2.1 *Detecting hallucination with cosine similarity*

From figure 3, it is evident that hallucination is the second most significant issue identified during the labeling process. Although research on hallucinations [37] provides a classification and understanding of their causes, it also highlights the lack of a robust quantitative method to accurately detect hallucinations. In this section, we aim to explore the relationship between hallucinations and their cosine similarities, and propose a potential method for detecting hallucinations by calculating cosine similarity.

The results are shown in Figure 5, which illustrates the differences in embeddings. **Ref vs Nor** represents the cosine similarity between the reference comment snippet and the normal comment snippet, while **Ref vs Hal** represents the cosine similarity between the reference comment snippet and the hallucinated comment snippet. A clear difference can be observed: the median similarity for Group1 is approximately 0.95, indicating a higher central tendency compared to Group2, which has a median of about 0.92. Furthermore, the interquartile range (IQR) is narrower for Group1, illustrating a more concentrated distribution of data points. In contrast, Group2 shows a wider IQR and extended lower whisker, highlighting a greater dispersion among data points. This phenomenon can be observed even when only one word in the comment changes to convey a hallucinatory meaning, while the rest of the comment and the code itself remain unchanged, as illustrated in Figure 6.

To substantiate the validity of our findings, we conducted Principal Component Analysis (PCA) on subsets of data that exhibited the largest and smallest cosine similarity discrepancies between the two groups in question. The outcomes are illustrated in Figure 7. For the data with the minimal cosine similarity, the dominant principal component (PC) was PC2, with negligible variation in PC1. Conversely, in the data with the maximal difference, the principal components for the Nor2 and Ref2 data points closely align, whereas the Hal2 data points distinctly manifest higher values along PC2.


Original Comment

检查用户输入是否为空并验证用户年龄

Translation: Check if user input is empty and verify user age

Predict Comment

检查用户输入是否为空并验证用户性别

Translation: Check if user input is empty and verify user gender

Figure 6: Example of original comment compared with hallucinatory predicted comment.

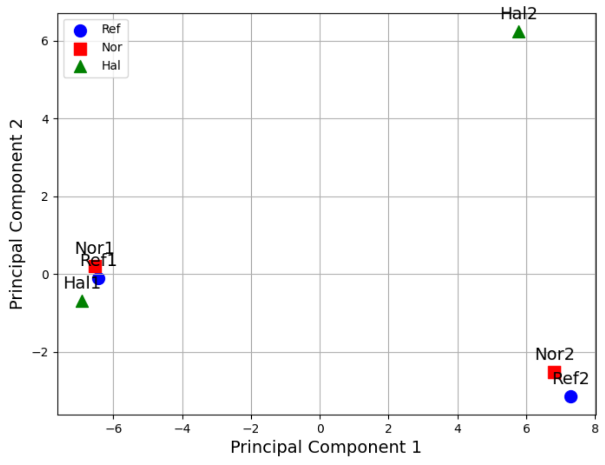


Figure 7: PCA results for two data examples.

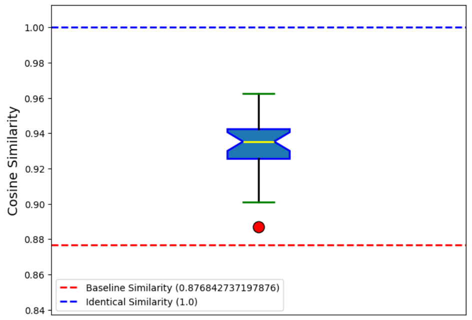
This analysis corroborates the hypothesized relationship between semantic content and word embedding dimensions.

4.2.2 *Using cosine similarities to differentiate between overly broad and normal comments*

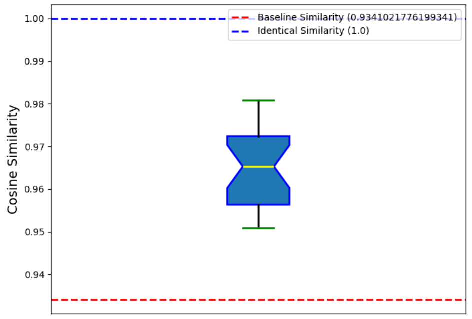
Figure 8 presents the box plot results for cosine similarity analyses, distinguishing between line comments 8a and block comments 8b.

The results depicted in Figure 8a demonstrate that the cosine similarities are tightly clustered around a median value of approximately 0.935, with the values tightly clustered around this median. The interquartile range (IQR) indicates that most values fall between about 0.932 and 0.938, suggesting a high degree of semantic similarity with the reference comments analyzed. This indicates that while the differences between overly broad comments and specific reference comments are subtle, they are statistically significant when compared to the perfect similarity score of 1.0 observed for identical comments and the baseline for semantic destroyed comments. This distinction is critical as it highlights the measurable semantic disparity even in cases where differences may initially appear minimal. Moreover, similar patterns were observed in the analysis of block comments, as presented in Figure 8b.

This consistent difference across different types of comments underscores the point that even seemingly minor deviations in text can lead to significant changes in semantic interpretation. Because of this difference, it is feasible to use cosine similarity to distinguish



(a) Box plot of line comment cosine similarity analysis.



(b) Box plot of block comment cosine similarity analysis.

Figure 8: Box plot result for cosine similarity analysis.

overly broad content from normal content when embeddings can be generated correctly, which answers our second research question.

5 Discussion

This study set out to investigate two key questions related to the performance and reliability of LLMs in generating Chinese inferences for Java code comments. First, we examined the types of errors that LLMs commonly make in this context. Second, we explored the potential of using cosine similarity as a quantitative method for analyzing semantic errors in generated comments. he following sections discuss our findings in detail, highlighting their significance and implications for future research.

5.1 Errors Made by LLMs in Chinese Inference on Java Code Comments

In our study, we identified several types of errors that large language models (LLMs) make when generating Chinese inferences for Java code comments. These errors can be broadly categorized into Model Specific Errors, Linguistic Errors, Semantic Errors, Syntax Errors, and Miscellaneous Errors. From the previous qualitative analysis, we can find that the large language model, here stable-code-3b, has obvious concentration in the types of errors when generating Chinese comments. In our study, the most common errors were semantic errors, followed by model specific errors, and then some syntax errors, while grammar errors were not manually detected during our labeling process. Such error distribution reflects that when the model deals with code comment generation problems in non-English languages, in this case Chinese, the error distribution generated has common errors in all non-English languages, and also generates language-specific errors according to different languages.

As for the lack of grammatical errors in the research process, we can attribute it partially to the particularity of the Chinese language we studied. Unlike most Western Germanic languages, the Classical Chinese [38] currently in use, is characterized by its lack of inflection, meaning that most words maintain a single grammatical form. Consequently, this linguistic feature leads to an absence of tense in Chinese [39]. Furthermore, the distinctive nature of Chinese characters greatly reduces the likelihood of erroneous language detection during inference processes. Additionally, Chinese functions as a mono-morphemic language, where each character represents an entire word equivalent to English, thus eliminating common errors found in other languages, such as extraneous letters following correct characters. These unique attributes provide an explanation for the absence of entries under the **Linguistic Errors** section in our taxonomy result for Chinese, whereas this category frequently encapsulates numerous errors in other language analyses.

From a linguistic perspective, the presence of semantically trivial errors and the absence of grammatical errors in the labeling process become explainable due to the high complexity [40] and low grammatical sensitivity of Chinese. From the perspective of the model, stable-code-3b is a large language model with only three billion parameters, and its implementation is based on decoder-only. Also, Chinese is not supported natively during the original training process. Therefore, we have reason to speculate that more problems will arise at the model level.

One of the main results of this study is the proposal of a complete Error Taxonomy for Evaluation of LLMs on code for Chinese use-cases. Through multiple rounds of optimization by multiple people in the open coding process, this taxonomy has gradually been able to cover most of the problems that may arise in generating annotations for large models in a multi-language environment. In the current field of evaluating the multilingual performance of large models, there is no similar work to evaluate the performance of large language models in generating code comments in non-English environments. Therefore, the significance of this work lies in that we propose a new error taxonomy that can play a guiding role in this field.

Existing LLM performance benchmarks usually focus on evaluating the performance of models on general tasks, such as natural language processing tasks, translation, or summary generation, etc. These benchmarks lack in-depth analysis of domain-specific applications, especially in the highly specialized task of code comment generation. Moreover, most of these evaluation criteria are based on English usage scenarios to evaluate performance, which does not meet our expectations for improving the performance of large language models in non-English usage scenarios. By building an error classification system specifically for Chinese code comments, we are able to more accurately identify the weaknesses and limitations of the model in this specific task. This not only helps to understand the shortcomings of existing models, but also provides a clear direction for future model improvements. For example, by analyzing common semantic errors and model-specific errors, we can develop more effective training strategies and data augmentation methods to improve the practical application effect of the model. As we said at the beginning, the significance of this study is to bridge the gap between non-English use cases and LLM for code comment generation.

5.2 Analyzing Semantic Errors via Cosine Similarities of Word Embeddings

For the second research question, through our previous quantitative analysis, we have verified the possibility of analyzing semantic errors by cosine similarity.

When conducting this research, several types of semantic errors were identified, with "Too General" being one of the most prevalent. This category denotes results that are vague and unclear. By conducting cosine similarity analyses on the embedding vectors of overly broad annotations compared to specific reference annotations, we can quantitatively assess their semantic similarities and differences [41]. This analysis aids in pinpointing which broad annotations are semantically aligned with the specific reference annotations and which ones are markedly divergent. The overly broad comments, which lack specificity and detail, were hypothesized to exhibit different cosine similarity patterns compared to more precise and contextually appropriate comments. Our results indicated that overly broad comments tend to have a lower cosine similarity to a wide range of reference comments. Although they are more general, these comments are not helpful for understanding the code. In contrast, normal comments, which are more tailored and specific, showed higher cosine similarity to the exact reference comment. This differentiation suggests that cosine similarity can serve as a tool for assessing the specificity and relevance of generated comments.

Our research also explored the feasibility of detecting hallucinations in LLM-generated comments using cosine similarity. By comparing the cosine similarities between reference comments and both normal and hallucinated comments, we found a significant distinction. Hallucinated comments show lower cosine similarity to the reference comments compared to normal comments written by senior programmer. This suggests that cosine similarity is a viable metric for identifying hallucinations, as the lower similarity scores reflect a divergence from the intended meaning or content of the reference, which is also verified by our PCA experiment. This method provides a quantitative approach to hallucination detection, which has been a challenge in the field due to the subjective nature of evaluating generated content. The ability to measure hallucinations quantitatively offers a promising direction for improving the reliability of LLMs in generating accurate and relevant code comments.

By establishing the relationship between the cosine similarity of word embeddings and semantic errors, we can describe semantic errors in a quantitative way. This not only establishes clearer standards for error analysis, butt also reduces human bias. At the same time, verifying the relationship between the two also proves the possibility of automating this judgment process. Promoting such a method can greatly reduce the time required to improve the Error Taxonomy, thereby improving the predictability of LLM performance.

6 Future work

In this project, our initial success with the Chinese dataset has been a springboard for a broader, more comprehensive approach to analyzing comment generation and error taxonomy across multiple languages. We have developed a robust methodology that not only yields insights into Chinese language performance but is also adaptable for assessing other languages. Currently, we are applying this refined analysis technique to Dutch, Polish, and Greek datasets, with the expectation of achieving similar success.

This strategic expansion allows us to explore the unique challenges posed by each language, enriching our understanding of linguistic nuances and model performance across diverse linguistic settings. By standardizing our analytical approach, we ensure consistent quality and comparability in our findings, which is crucial for developing better multilingual natural language processing tools. Our work continues to focus on refining these methods to further enhance the accuracy and relevance of our models, thereby supporting a broader application and better performance across various languages.

7 Conclusion

In this study, we aimed to investigate two key questions regarding the evaluation of LLMs on code for Chinese use-cases. Firstly, we identified the common types of errors LLMs make, leading to the development of a detailed error taxonomy. This taxonomy revealed that semantic errors, particularly hallucinations, are prevalent, while linguistic errors are less common due to the unique characteristics of the Chinese language.

Second, we validate the feasibility of using cosine similarity of word embeddings to analyze semantic errors. Code comments with semantic errors usually have lower word embedding cosine similarity than regular comments, explaining their larger deviation in semantics. The significance of this study lies not only in finding the connection between semantic errors and cosine similarity, but more importantly, it proposes a possible direction for automatically analyzing the performance of large language models.

These findings have significant implications for the development and improvement of LLMs, particularly in generating accurate and relevant code comments in Chinese. By providing a detailed error taxonomy and demonstrating the utility of cosine similarity in detecting hallucinations and evaluating comment specificity, this study contributes valuable insights to the field.

However, our research has certain limitations, including the specific focus on code comments and the reliance on cosine similarity as the primary metric. Future research should explore additional languages and alternative metrics to build on our findings. Moreover, expanding the dataset and labeling more comments could further enhance the universality of our error taxonomy.

In conclusion, this study provides a thorough analysis of the errors made by LLMs in Chinese code commenting and introduces innovative methods for improving the analyzation of semantic errors in generated comments. These contributions lay the groundwork for future advancements in applying LLMs to multilingual code annotation tasks, ultimately enhancing their practical utility and reliability.

8 Ethics and Responsible Research

For the completion of our research, we utilized two advanced models: `stable-code-3b` and `Ernie 1.0`. These models are governed by the Stability AI Non-Commercial Research Community License and the Apache-2.0 license, respectively. These licensing terms allow for the application of these models in academic and scientific research, ensuring compliance with legal and ethical standards.

To gather the necessary data for our dataset, we sourced information exclusively from open-source repositories available on GitHub. This approach ensures transparency and accessibility of our data sources, enabling other researchers to verify and build upon our work. Utilizing open-source repositories not only aligns with ethical standards but also Spromotes the integrity of ourSSS research by providing a clear and accessible trail for data verification.

Ensuring the reproducibility of our research findings is crucial. To this end, we have established a public GitHub repository, which can be accessed at GitHub Repository. In this repository, we have disclosed the repositories from which we collected data, the raw results of our tertiary study, and all the code utilized in our analyses. The repository also includes detailed documentation to guide other researchers through our processes and methodologies, ensuring they can replicate our study accurately.

Furthermore, we have uploaded the dataset, which includes the leading comments, to a publicly accessible repository at Hugging Face. This enables other researchers and practitioners in the field to access our dataset readily, facilitating further research and application in related fields. By making both the data and the code publicly available, we encourage transparency and foster trust in our findings, as every aspect of our research can be scrutinized and validated by the scientific community.

We also took careful measures to ensure the integrity of our data and the ethical considerations involved in its use. All data collection processes adhered to the guidelines for ethical research, respecting the licenses and usage terms of the sources. By using only open-source data, we avoided potential legal and ethical issues related to proprietary or sensitive information.

In addition to these technical and ethical measures, we conducted a thorough reflection on the challenges of reproducibility and integrity in our research. We recognize that reproducibility is not just about making data and code available; it involves ensuring that other researchers can understand and implement our methodologies. To this end, we provided comprehensive documentation and detailed explanations of our experimental setup, data processing steps, and analysis techniques. This level of detail helps mitigate common reproducibility challenges, such as unclear methodologies or missing information, which can hinder other researchers from successfully replicating studies.

By taking these steps, we aim to foster an environment of open scientific communication and collaboration, allowing the community to benefit from our findings while adhering to the highest standards of research integrity and reproducibility. Our commitment to these principles not only enhances the credibility of our research but also contributes to the advancement of knowledge in the field, providing a solid foundation for future studies to build upon.

Acknowledgments

Finally, I have the chance to write down my feelings for the past three years. For this universe, we are just travellers staying shorter than a blink. For ourselves, the world is everything, occupying our whole lifetime. It's my great honor to study in Delft University of Technology, one of the **best** technical universities in the world. I always believe the theory of "No pain, No gain", and I'd like to say for all the pain I suffered during my study, it worth. I hope this paper could be useful to the rest of the world, contributing my value, but not just creating a 1MB file in the database.

There's lots of people I'd like to share my acknowledgement with. First, I have to show my respect to myself. Surviving from one of the hardest Bachelor education system is my lifelong honor and medal. Then, I'd like to gratitude my parents, especially my mother Dan Gao. She can always be my strongest back shield and help me get out of difficulties. Moreover, I want to thank all my tutors that helped me reach the place I have today, especially my supervisor for this paper ir. Jonathan Katzy and Assistant Prof. Dr. Maliheh Izadi. Without their help, I cannot grow up from an infant to an engineer. Finally, I have to give special acknowledgement to my friends. I have discovered lots of truly marvellous stories among us, which this margin is too narrow to contain. Therefore, I decide to put a name list here: Chen Yiming, Cui Minghui, Gu Shengfei, Han Xingyu, Hu Chen, Huang Chengcheng, Sun Xiaoxi, Tang Shenyi, Wang Tingyue, Yang Renyi, Zhang Yechen

References

[1] Albert Ziegler, Eirini Kalliamvakou, Shawn Simister, Ganesh Sittampalam, Alice Li, Andrew Rice, Devon Rifkin, and Edward Aftandilian. Productivity assessment of neural code completion, 2022.

[2] Steven I. Ross, Fernando Martinez, Stephanie Houde, Michael Muller, and Justin D. Weisz. The programmer's assistant: Conversational interaction with a large language model for software development. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23. ACM, March 2023.

[3] Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. Programming is hard – or at least it used to be: Educational opportunities and challenges of ai code generation, 2022.

[4] Alibaba Cloud. Alibaba cloud pilots ai coding assistant to help employees write code, 2023. Accessed: 2024-05-26.

[5] Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual BERT? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001, Florence, Italy, July 2019. Association for Computational Linguistics.

[6] Microsoft. Microsoft research project helps languages survive and thrive. https://news.microsoft.com/source/asia/features/microsoft-research-project-helps-languages-survive-and-thrive/#:~:text=This%20means%2088%20percent%20of,to%20navigate%20the%20digital%20world, 2024. [Online; accessed 8-June-2024].

[7] Xiang Zhang, Senyu Li, Bradley Hauer, Ning Shi, and Grzegorz Kondrak. Don't trust chatgpt when your question is not in english: A study of multilingual abilities and types of llms, 2023.

[8] Lingfeng Shen, Weiting Tan, Sihao Chen, Yunmo Chen, Jingyu Zhang, Haoran Xu, Boyuan Zheng, Philipp Koehn, and Daniel Khashabi. The language barrier: Dissecting safety challenges of llms in multilingual contexts, 2024.

[9] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR*, abs/2102.04664, 2021.

[10] Qinkai Zheng, Xiao Xia, Xu Zou, Yuxiao Dong, Shan Wang, Yufei Xue, Zihan Wang, Lei Shen, Andi Wang, Yang Li, Teng Su, Zhilin Yang, and Jie Tang. Codegeex: A pre-trained model for code generation with multilingual evaluations on humaneval-x, 2023.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[12] Timo Pawelka and Elmar Juergens. Is this code written in english? a study

of the natural language of comments and identifiers in practice. In *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 401–410, 2015.

[13] Nikhil Pinnaparaju, Reshinth Adithyan, Duy Phung, Jonathan Tow, James Baicoianu, and Nathan Cooper. Stable code 3b.

[14] Faisal Rahutomo, Teruaki Kitasuka, Masayoshi Aritsugi, et al. Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1. University of Seoul South Korea, 2012.

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics.

[16] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[17] Junayed Mahmud, Fahim Faisal, Raihan Islam Arnob, Antonios Anastasopoulos, and Kevin Moran. Code to comment translation: A comparative study on model effectiveness & errors. In Royi Lachmy, Ziyu Yao, Greg Durrett, Milos Gligoric, Junyi Jessy Li, Ray Mooney, Graham Neubig, Yu Su, Huan Sun, and Reut Tsarfaty, editors, *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, pages 1–16, Online, August 2021. Association for Computational Linguistics.

[18] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), mar 2023.

[19] Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. *arXiv preprint arXiv:1704.07138*, 2017.

[20] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

[21] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization, 2020.

[22] Will Lowe. Towards a theory of semantic space. In *Proceedings of the annual meeting of the cognitive science society*, volume 23, 2001.

[23] Yasin Abbasi Yadkori, Ilja Kuzborskij, András György, and Csaba Szepesvári. To believe or not to believe your llm, 2024.

[24] Peipei Xia, Li Zhang, and Fanzhang Li. Learning similarity with cosine similarity ensemble. *Information Sciences*, 307:39–52, 2015.

[25] Junliang Luo, Tianyu Li, Di Wu, Michael Jenkin, Steve Liu, and Gregory Dudek. Hallucination detection and hallucination mitigation: An investigation, 2024.

[26] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM on Web Conference 2024*, pages 887–890, 2024.

[27] Ansar Aynetdinov and Alan Akbik. Semscore: Automated evaluation of instruction-tuned llms based on semantic textual similarity, 2024.

[28] oprogramador. Most common words by language, 2024. Accessed: 2024-05-26.

[29] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, and et al. Gemma. 2024.

[30] AISE-TUDelft. LLM of Babel ZH2. https://huggingface.co/datasets/AISE-TUDelft/LLM-of-Babel-ZH2, 2023.

[31] D4vidHuang. Llm of babel zh2 rndselection new. https://huggingface.co/datasets/D4vidHuang/LLM_Of_Babel_ZH2_RNDSelection_new, 2024. Accessed: 2024-06-02.

[32] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. Efficient training of language models to fill in the middle, 2022.

[33] Shahedul Huq Khandkar. Open coding. *University of Calgary*, 23(2009):2009, 2009.

[34] Julianne S Oktay. *Grounded theory.* Oxford University Press, 2012.

[35] Streamlit. https://github.com/streamlit/streamlit. Accessed: 2023-06-03.

[36] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*, 2019.

[37] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, 2023.

[38] Edwin George Pulleyblank. *Outline of classical Chinese grammar.* Ubc Press, 1995.

[39] Jo-Wang Lin. Time in a language without tense: The case of chinese. *Journal of semantics*, 23(1):1–53, 2006.

[40] Haitao Liu. The complexity of chinese syntactic dependency networks. *Physica A: Statistical Mechanics and its Applications*, 387(12):3048–3058, 2008.

[41] Jonathan Katzy, Maliheh Izadi, and Arie van Deursen. On the impact of language selection for training and evaluating programming language models. In *2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 271–276. IEEE, 2023.

A Complete PCA diagram

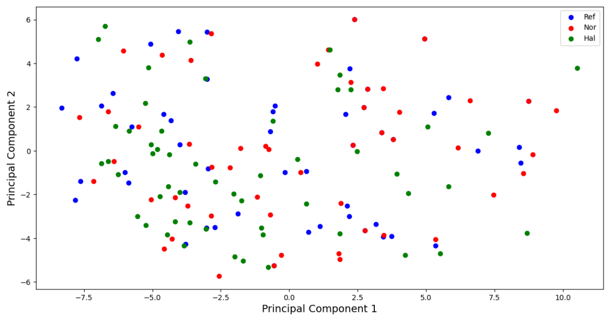


Figure 9: Complete PCA diagram for all data.

B Complete cosine similarity bar chart

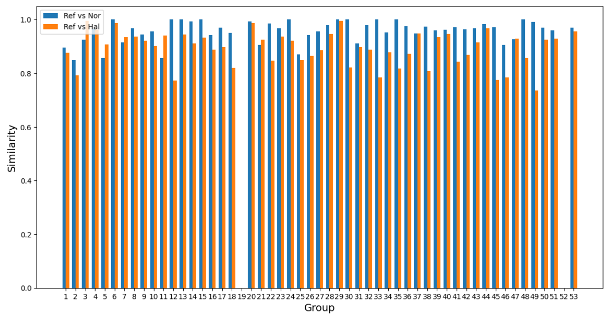


Figure 10: Complete cosine similarity bar chart for all data.

The reason we leave data No.19 and No.52 blank is because the senior programmer thinks no comments are needed here.

C Use of Large Language Models in the Research Project

In the construction of this research paper, AI tools are utilized, specifically LLMs like ChatGPT, to assist in refining the grammar and checking of the spelling. This application was limited to grammatical corrections and enhancing the readability of the paper without altering the conceptual content or analytical insights derived from my research.