LLM of Babel: An analysis of the behavior of large language models when performing Java code summarization in Dutch

Gopal-Raj G. S. Panchu
EEMCS, Delft University of Technology, The Netherlands

Supervisor(s): 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: Gopal-Raj G. S. Panchu
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Arie van Deursen , Assistant Prof. Dr. Maliheh Izadi, Assistant Prof. Dr. Gosia Migut, Ir. Jonathan Katzy

An electronic version of this thesis is available at http://repository.tudelft.nl/.

LLM of Babel: An analysis of the behavior of large language models when performing Java code summarization in Dutch

Gopal-Raj G.S. Panchu
Delft University of Technology
Delft, Netherlands

ABSTRACT

How well do large language models (LLMs) infer text in a non-English context when performing code summarization? The goal of this paper was to understand the mistakes made by LLMs when performing code summarization in Dutch. We categorized the mistakes made by CodeQwen1.5-7b when inferring Java code comments in the Dutch language through an open coding methodology to create a taxonomy of errors by which to categorize these mistakes.

Dutch code comments scraped from Github were analyzed, resulting in a taxonomy that revealed four broad categories under which inference errors could be classified: Semantic, Syntactic, Linguistic, and LLM Specific. Additional analysis revealed a prevalence of semantic and LLM specific errors in the dataset compared to the other categories. The resulting taxonomy has significant overlap with other taxonomies in similar fields like machine translation and English code summarization while introducing several categories that are not prevalent in those fields. Furthermore, it was found that BLEU-1 And ROUGEL metrics were unreliable as accuracy measures in this use case due to their nature as similarity metrics.

KEYWORDS

Automatic Code Completion, Transformers, Language Models, Open Coding, Multilingual,Taxonomy, Code summarization

1 INTRODUCTION

Over the last few years, the usage of Large Language Models (LLMs) has increased significantly, and uses in computer science have become increasingly common [37]. Existing research addresses the performance while using these models in the context of different natural languages [26]. Additionally, it has been researched how these models behave when given coding assignments in different languages [33]. The consistent result seems that LLMs perform better in English than many other languages. However, the performance of LLMs when summarizing code by way of code comments in non-English languages remains largely unexplored.

This paper aims to clarify how LLMs perform when summarizing Java code in Dutch, as a way to broaden our knowledge about multilingual LLM performance. To evaluate the performance of LLMs in this area, a taxonomy of errors is required. In this paper, such an error taxonomy will be created through an open coding methodology. Afterward, the overlap of this error taxonomy with related fields will be analyzed. The goal of this paper is to understand the mistakes made by LLMs when performing code summarization in Dutch which leads to the following research questions:

RQ1: What are the most common mistakes made by LLMs in the context of Java code summarization in the Dutch language?

RQ2: Are BLEU-1and ROUGE-L metrics reliable indicators for the accuracy of code summarization inferences?

RQ3: To what extent does the multilingual code summarization error taxonomy overlap with existing error taxonomies in machine translation and code summarization?

These questions will be explored by analyzing the performance of CodeQwen 1.5-7B [30] when inferring Dutch Java comments. CodeQwen1.5-7B was chosen as it is a new model made by a large company with a focus on performance in multilingual tasks while also being of reasonable size. This allows for a large amount of inferences using limited resources. Our contributions are as follows:

- Creation of an error taxonomy by which mistakes made by an LLM can be classified in the context of multilingual code summarization.
- Labeling of the mistakes made by an LLM when generating code documentation in Dutch, leading to an analysis of the major mistakes LLMs make in the context of the Dutch language while also taking multilingual contexts into account.
- Comparison of the error taxonomy for non-English comment inference to taxonomies in similar fields.
- Highlighting of the contexts in which these LLMs seem to already perform well.

2 RELATED WORKS

This paper finds itself at the intersection of various larger fields of research, so a selection of related works will be introduced in this section to clarify the background of the subject matter. At first, we will introduce the area of multilingual LLMs to create an understanding of the technology we will be analyzing. After which we will be elaborating upon works that touch upon the area of automatic code summarization. Finally, we will touch upon the open coding methodology which was used for the quantitative analysis aspect of this paper.

2.1 Multilingual LLMs

The multilingual usage and application of LLMs has been researched [17, 20, 29, 35, 38]. LLMs are limited by the data that was used to train the models, which may create a bias towards any trends that occur in the underlying data[9, 11]. Modern LLMs are usually trained on datasets that were scraped from the internet and English performs better than other languages on average[29, 36]. As stated earlier, the field of multilingual LLMs has been garnering more attention which has led to more attention on languages other than

English. The fact remains however that despite recent improvements most models will not perform as well in languages other than the languages which take up a majority of the training dataset.

2.2 Automatic code summarization

Automatic code comment generation encompasses the action of effectively summarizing code into a human-readable format through automated means. Correct code documentation has benefits for a code base, such as easier maintenance and improved comprehension [7, 10]. Through automation, it is possible to leverage these benefits while limiting the time investment needed for quality documentation. There is existing research regarding the generation of automatic code summarization[16, 23]. Even for a relatively new field such as Large Language Models, research has been done on the effectiveness of these programs [1, 2, 21]. Most research uses English code summarization however, which leaves a knowledge gap regarding automatic code summarization in languages other than English. It seems that there is less research regarding error taxonomies, as there are few papers about the subject. The SCATE[31] and MQM[22] taxonomies which are translation taxonomies are widespread and seem to be used if a taxonomy base is needed.

2.3 Opencoding

Open coding is an analytic process that uses labels to attach concepts to observed data during qualitative analysis. This method attempts to classify data into meaningful segments which naturally leads to its usage to create a taxonomy[24]. In the context of code-related error taxonomies, this methodology has been used before[19, 21].

3 APPROACH

Java code files containing Dutch comments were scraped from Github, resulting in a dataset that was suitable for this research. After which any files that did not adhere to our exclusion criteria were filtered out. Code comments within these files were then found using regular expressions and automatically analyzed to determine whether they contained Dutch. The comments were then span-masked and inferred by an LLM with the rest of the file as context. The inferred comments were analyzed with the goal of extracting errors. Quantitative analysis was used to evaluate the similarity of the comments to their originals. After quantitative analysis, we performed qualitative analysis by manually classifying errors according to an open coding approach to create a taxonomy of errors. Finally, we proceeded to compare the created taxonomy to other taxonomies of errors in similar fields.

3.1 Quantitative

It was decided to look at the BLEU[25] and ROUGE[18] metrics of the inferred comments in comparison to the original comments to increase our understanding of their similarity. These scores generally measure similarity in the context of translation and summarization respectively. As code summarization can be seen as a translation and summarization task we decided to use these metrics for quantitative analysis. We used the original comment as the reference and the inferred comment as the candidate for every score calculation. BLEU1 was used as our BLEU metric as other BLEU metrics would not work correctly with shorter sequences due to their weight allocations. For ROUGE analysis we used ROUGEL with a separate generation of its precision, recall, and F1 scores. We decided to focus on the F1 score as this is a combination of precision and recall.

3.2 Qualitative

We classified all errors within comments with as much detail as possible initially, in order to aid in taxonomy creation. An initial subset of 200 comments was selected for analysis. The generated comments were manually compared to the originals to classify any possible errors according to a hypothetical error taxonomy. In addition, the accuracy of the comments was recorded using three labels:

- Inaccurate (0), this label was used in case the inferred comment was inaccurate in relation to the relevant code.
- Partially accurate (1), this label was used if the inferred comment contained accurate or partially accurate information but would not be usable without manual correction.
- Fully accurate (2), this label was used if the inferred comment was deemed usable as is.

A candidate error taxonomy was refined using the results of the initial qualitative analysis. This evaluation process was repeated using another subset of 200 comments and a final subset of 300 comments. After every repetition, we discussed our findings regarding the error taxonomy and adapted the taxonomy to reflect any changes that needed to be made according to newly found categories. While adapting the taxonomy, inclusion and exclusion criteria were defined per category, to clarify when an error was to be labeled within them. Finally, comments were labeled using the final taxonomy to create a set of 600 comments to perform further analysis[1].

Exclusion criteria for the dataset were defined to ensure the relevance of the collected data in relation to the research questions:

- Comments containing licensing information were excluded due to their lack of semantic relevance to the code.
- Comments with less than three words were excluded as these comments would be unmasked due to our context-saving measures.
- Comments that did not contain Dutch were excluded.
- Comments that mainly contained code were excluded as they do not contain natural language.

3.3 Taxonomy comparison

After finalizing the taxonomy of errors, we found taxonomies in similar fields to evaluate their similarity to the taxonomy that resulted from this experiment.

Two research databases were used to find error taxonomies in the fields of machine translation and code summarization, both of which were accessed on 19/06/2024. These databases were chosen due to their smaller sizes, which allowed for a rigorous research methodology within the time constraints:

- Scopus
- Web of Science

[1]https://huggingface.co/datasets/NovoGSP/LLM-of-Babel-NL-Labeled

Due to rapid improvements in the field of LLMs, the search space was limited to articles published in the last three years.
The following search queries were used:

"error taxonomy" OR "error Category" OR "error categories" OR "taxonomy of errors" AND "code summarization" OR "code summation" OR "code documentation" OR "code comments" AND "open coding"

"error taxonomy" OR "error category" OR "error categories" OR "taxonomy of errors" AND "translation" AND "LLM" OR "large language model*" OR "transformer" OR "artificial intelligence" OR "machine learning"

error OR mistake OR failure AND taxonomy OR "open coding" AND "automatically generated" OR "machine translation" OR "summarization"

After removing duplicates, a total of 45 papers were found. These 45 papers were then read and checked for the presence of the following criteria:

- An error taxonomy generated with the open coding methodology has to be present.
- The main focus of the paper needs to be on machine translation or code summarization involving natural language.

Following this analysis, 3 papers remained that adhered to all requirements. The error taxonomies from these papers were analyzed in relation to the taxonomy which resulted from our qualitative research to identify any overlaps and differences.

- Code to Comment Translation: A Comparative Study on Model Effectiveness & Errors (Mahmoud er al.) [21]
- A Survey of Recent Error Annotation Schemes for Automatically Generated Text (Huidrom and Belz) [13]
- Towards a Consensus Taxonomy for Annotating Errors in Automatically Generated Text (Sharou and Specia) [28]

These taxonomies were compared through an analysis of the leaf nodes of each taxonomy in comparison to the taxonomy in this paper. If a leaf node of the taxonomies could be classified under one of our leaf nodes or was identical it was labeled as such. Otherwise, a leaf node was labeled as non-overlapping.

4 DATA

4.1 Dataset

The research dataset[2] was created by scraping GitHub for repositories containing the 2,500 most common words in Dutch and selecting the top 100 files according to the Github search based on those words. This resulted in 139,488 eligible code files after removing duplicates. We proceeded to filter out any files that did not contain any comments by searching for the existence of comment tags, resulting in 120,926 eligible files. The Qwen1.5 tokenizer was run on the remaining files, which allowed filtration of the dataset with a maximum token length of 8192, bringing the files down to 103,396. 1,037,750 comments were identified within these files using regular expressions, which were reduced down to 85,912 eligible comments after filtering out any comments that were deemed to not be Dutch through the langdetect library [3]. To limit the influence a particular repository can have on the error distribution, it was decided to take a single comment per repository chosen randomly to create the final dataset consisting of 4,826 comments. To summarize, we filtered the original dataset in the following criteria:

- The files had to contain comments.
- The files had to have a maximum length of 8192 tokens.
- The comments had to be Dutch.
- The files could not have overlapping repositories.

4.2 Comment Inference

One of the parameters necessary for inference is a limit on the number of new tokens that can be generated. A compromise between size and inference time was made regarding this parameter by finding the distribution of token lengths of all comments that were eligible for inference before filtering on the repository limit and setting the maximum to a number that would result in 95% inclusion of this dataset. What follows is the distribution of the 85,912 eligible comments.

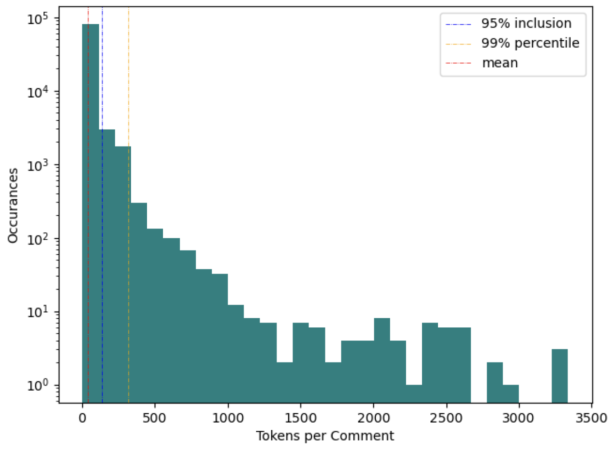

Figure 1: Tokens per comment

It can be seen that a majority of the comments are included with a maximum token length of 138. The logarithmic distribution causes a higher token length to result in increasingly worse inclusion percentages per token while increasing inference time per comment significantly. Every comment was found using regular expressions, with which the Java comment syntax was targeted to extract comments. Three words were left in the context in case of block comments and two words in case of line comments to encourage the LLM to predict in Dutch instead of defaulting to English predictions.

The DelftBlue [8] computing clusters provided by the Delft University of Technology were used to run the inference pipeline. Fill-in-the-middle[4] was used to give the model as much context as possible to ensure proper inference. Additionally, the maximum token length of these files was limited to 8,192 tokens. This is the maximum context for a lot of models, specifically codegemma2-7B[5]

[2]https://huggingface.co/datasets/AISE-TUDelft/LLM-of-Babel-NL2

[3]https://pypi.org/project/langdetect/

as it is one of the newer available models that has code inference capabilities. The initial set consisted of 85,912 comments, which was reduced to 4,826 comments after selecting a subset of comments while limiting overlapping repositories. These comments were then inferred through a huggingface pipeline [34] with CodeQwen1.5-7B [30] as the inference model.

4.3 Language

The CodeQwen1.5-7B model is based on the Qwen1.5 model, which is advertised as a multilingual model. It is mainly trained on English and Chinese according to the accompanying paper[30]. Linguistically speaking, the Dutch language has a lot of similarities to English due to not only their Indo-European roots but also their West Germanic origins [12, chapter 1]. English is an Ingvaeonic language, while Dutch has Istvaeonic origins [12, chapter 1]. Additionally, modern Dutch has many English influences due to the prevalence of the English language in the Netherlands, especially among youths[32]. This may be strengthened even further in the field of computer science, as many programming languages are based on English keywords.

Chinese languages consist of the Sinitic branch of the Sino-Tibetan language family[27], which means that it is not in the same Indo-European language family as English or Dutch. On top of the linguistic separation, a difference in writing systems between Chinese and Dutch may cause more problems in this case. These different writing systems create a situation in which any tokenizer will be unable to create tokens with overlap between the two languages, even if there were linguistic similarities. This naturally causes issues as the inherent impossibility of token overlap causes problems with semantic similarity.

5 RESULTS

5.1 Quantitative

We plotted the BLEU-1 and ROUGEL-F1 scores for the research dataset, resulting in the following graph which seems to follow a log distribution. A peak can be found at score 1 which is explained by identical matches between inferences and original comments. Another peak can be found at score 0 which is explained by inferences that have no overlap with their original comments.

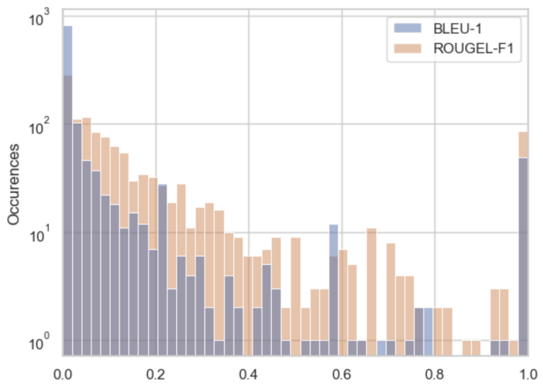

Figure 2: BLEU-1 & ROUGEL-F1 occurances

This graph shows that the inferred comments are either identical to the original or relatively dissimilar on average. Additionally, it shows the difference between BLEU-1 and ROUGEL-F1 scores, it seems as if the ROUGEL-F1 score is higher on average. This can be explained by the difference in calculations where ROUGEL-F1 evaluates the longest common subsequence whereas BLEU-1 evaluates direct similarity.

The BLEU-1 and ROUGEL-F1 scores were also plotted against the accuracy for the 600 manually analyzed comments. Boxplots were used to analyze the relation between these two components. It can be seen that for both the BLEU-1 and ROUGEL-F1 metrics there are low mean scores except for cases with Accuracy 2, this indicates that on average fully accurate inferences have a higher BLEU-1 and ROUGEL-F1 score. In addition, it can be seen that the BLEU-1 and ROUGEL-F1 scores of partially accurate inferences do not differ much from their inaccurate counterparts.

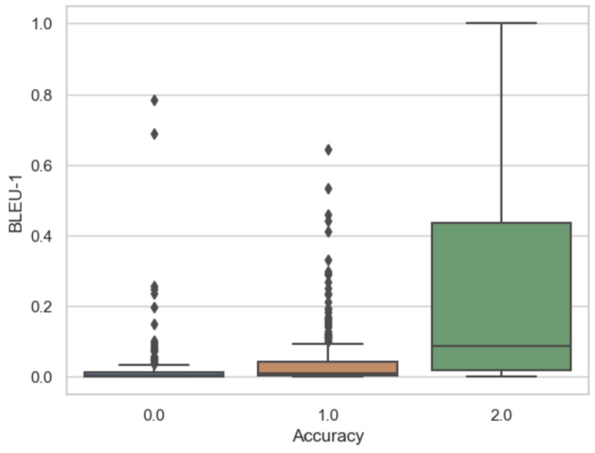

Figure 3: BLEU-1 compared to Accuracy

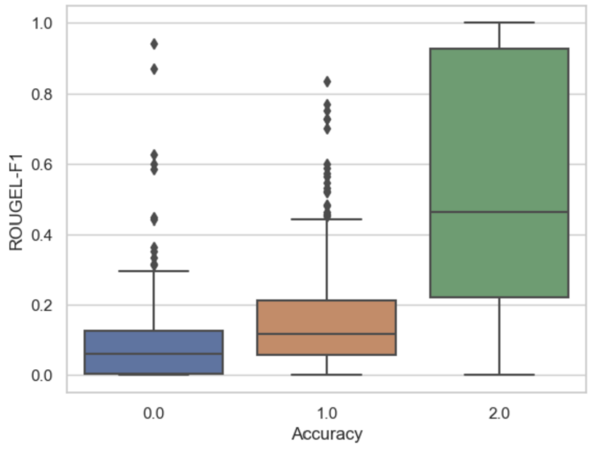

Figure 4: ROUGEL-F1 compared to Accuracy

5.2 Qualitative[4]

Table 1: Taxonomy of errors including occurrence count and number of overlapping leaf categories in compared taxonomies

Failure category plus label ID	Count	Mahmud et al.[21]	Sharou and Specia [28]	Huidrom and Belz[13]
MS Model-oriented Errors	**398**	-	-	-
└ MS-IG Incoherent Generation	3	-	-	1
└ MS-CC Copy context	58	-	-	2
└ MS-ME Memorization	13	-	-	-
└ MS-ME1 PII	9	-	-	-
└ MS-ME2 URL	3	-	-	-
└ MS-ME3 Training Memorization	1	-	-	-
└ MS-ET Early Termination	50	8	-	-
└ MS-LT Late Termination	107	-	-	1
└ MS-NG No Generation	0	-	-	-
└ MS-RE Repetition	167	-	-	-
└ MS-RE1 Pattern Repetition	57	2	-	1
└ MS-RE2 Verbatim Repetition	110	2	-	1
LG Linguistic Error	66	-	-	-
└ LG-GR Grammar	33	2	-	-
└ LG-GR1 Plurality	1	-	-	-
└ LG-GR2 Conjugation	2	-	-	-
└ LG-GR3 Gender	6	-	-	-
└ LG-GR4 Language Syntax	7	-	-	-
└ LG-GR5 Capitalization	1	-	-	-
└ LG-GR6 Cohesion	16	-	1	3
└ LG-IS Incorrect synonym	0	-	-	2
└ LG-WL Wrong language	33	-	-	-
└ LG-WL1 Undesired translations	5	-	1	1
└ LG-WL2 Incorrect language	28	-	1	1
SE Semantic error	**520**	-	-	1
└ SE-MD Missing Details	38	15	1	1
└ SE-TS Too specific	16	3	-	-
└ SE-HA Hallucination	290	-	-	-
└ SE-HA1 Misplaced Facts	33	1	3	1
└ SE-HA2 Out of Context	21	1	2	1
└ SE-HA3 In context	236	3	3	1
└ SE-CS Completion includes code	185	-	-	-
└ SE-CS1 Code commented out	30	-	-	1
└ SE-CS2 Code intended to run	155	-	-	1
ST Syntax	**8**	-	-	-
└ ST-IF Incorrect comment format	8	-	-	-
└ ST-IF1 Comment Syntax	1	-	-	-
└ ST-IF2 Omitted Identifier	7	-	-	-
+ Fully accurate	**131**			
E Excluded	**620**			

[4]This taxonomy was made in collaboration with 4 other students and takes Polish, Greek and Chinese examples into account as well.

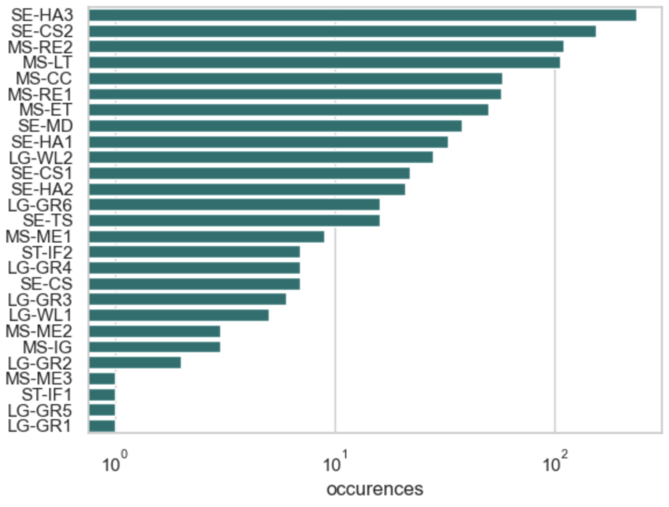

Figure 5: Label occurences

In addition to the occurrences of individual errors we found the error pairs which occurred together most frequently to be able to analyse the co-correlation between errors.

Error pair	Occurences
(SE-HA3, SE-CS2)	91
(SE-HA3, MS-LT)	52
(SE-HA3, MS-RE2)	43
(SE-CS2, MS-LT)	42
(SE-HA3, MS-RE1)	38
(SE-CS2, MS-CC)	27
(MS-LT, MS-RE2)	26
(MS-ET, MS-RE2)	22
(SE-HA3, MS-CC)	20
(MS-LT, MS-CC)	17
(SE-HA3, LG-WL2)	16
(MS-CC, MS-RE2)	11
(SE-HA3, SE-CS1)	11

Table 2: Error pair occurences

Additionally, we evaluated the occurrences of different accuracy values resulting in the following values:

- Inaccurate (0) - 182 occurrences
- Partially accurate (1) - 287 occurrences
- Fully accurate (2) - 131 occurrences

6 DISCUSSION

6.1 Quantitative analysis

We found that the BLEU-1 and ROUGEL-F1 scores of these inferences have a low mean value. The values also do not seem to always be directly related to the quality of a comment as they only reflect the similarity of an inference to its original comment. There are comments that are totally dissimilar to the original but of high accuracy, and comments that are highly similar to the original but of low accuracy. Several problems were identified which could explain these shortcomings. Both BLEU-1 and ROUGEL-F1 depend on the presence of reference sequences which are used to create the scores. As the dataset originated from open-source Github code there was a large disparity in quality, which resulted in unreliability in the reference sequences. Additionally, these scores perform better as a metric when there are several reference sequences, which are not easily obtainable in this use case. Besides these issues, there are often several correct ways to summarize code depending on the surrounding context.

It can be seen in Figure 2 that the ROUGEL-F1 score displays a higher peak at 1 than the BLEU-1 score, this may be explained by the tendency of models to keep generating new tokens even if the inference could be deemed sufficient causing comments with a higher recall score to occur which cause higher ROUGE-F1 scores.

6.2 Qualitative analysis

Our results in Table 1 point to four categories under which all errors can be classified, these categories have differing inclusion criteria justifying their separation. This naturally leads to an answer to **RQ1** as these four broad categories were used to classify all common mistakes made by LLMs within the provided context.

- Linguistic errors
- Semantic errors
- Model behavior errors
- Syntax errors

Linguistic errors are defined as errors that are related to the linguistic aspect of an inference. In this case, these errors were related to the Dutch language, it was found that most errors did not fall into this category.

Semantic errors are related to the meaning of a comment. An error was deemed as a semantic error if the fundamental meaning of an inference differed from the expectation in regard to the code context. These types of errors usually were accompanied by model behavior errors as can be seen from the following graph:

Model behavior errors are errors that are related to the known behavior of large language models. These errors are known as mistakes made by large language models and they illustrate the current weaknesses of these models within this context.

Syntax errors are related to the syntax of Java, they are errors that occur when the syntax of Java is broken or misused in some way resulting in an error. In the context of this paper, these errors were specifically interesting in cases where the Java comment syntax was broken or misused in some way.

Regarding the occurrences of specific errors, the following errors occurred most frequently according to Figure 5: SE-HA3, SE-CS2, MS-RE2, MS-LT, MS-RE1.

These errors point towards the ability of the model to operate within the correct context while making errors. The most common mistake made by the model was SE-HA3 which supports this claim. SE-HA3 indicates an in context hallucination of the model which indicates the usage of the surrounding context in a way that does not make sense for the inference itself.

Additionally, it seems as if these models oftentimes predict code when they should just be predicting comments. This can be seen by the prevalence of SE-CS1 and SE-CS2 errors.

In addition, MS-LT late terminations and MS-RE repetition were common mistakes. Both of which often occurred together with SE-HA3 errors. These mistakes are especially unfortunate as the late termination errors specify that the content before the additional inferences was accurate.

It also seems as if the model performs well in the context of Linguistic and Syntactic mistakes as these errors occurred relatively infrequently. The most common linguistic mistake was LG-WL2 which might be related to the linguistic similarity between Dutch and English as the incorrectly predicted language was always English. The second most common linguistic mistake was LG-GR6 which indicates an incoherent sentence, but this mistake only occurred 16 times.

Another point that should be addressed is the overall performance of the model on summarization tasks, our research indicated that the generated comment was oftentimes at least partially accurate. In fact, only 30.3% of generations were totally inaccurate according to the manual check. 131 Inferences were fully accurate and 287 inferences were at least partially accurate which means that they could be made accurate with adjustments.

6.3 Comparison

We found that our taxonomy contained almost all leaf nodes of the other taxonomies while sacrificing specificity. SE-MD "Missing Details" for example had overlap with 15 categories in the taxonomy created by Mahmoud et al[21]. This taxonomy was especially detailed regarding semantic errors, therefore all of the categories overlap with our taxonomy which contains relatively broader categories. This indicates that our taxonomy could possibly be expanded by adopting these categories as subcategories for their respective matches.

A focus point we would like to address is the categories that were not present in our taxonomy but were present in the other papers. The taxonomy created by Sharou and Specia[28] contained two categories that were not present in our taxonomy. One of these categories was "Deviation in toxicity (TOX)" which describes inference errors where the inferred text incites negative actions towards an individual or group based upon incorrect translation. We have not found any errors which can be categorized under TOX within our research dataset. The other category that was missing from our taxonomy was "Deviation in health/safety risks (SAF)" which describes inferences where health and safety risks are introduced into the text through translation errors. We also were unable to find any such occurrences within our dataset. We believe these categories to be important to consider nonetheless due to their negative impact when they do occur. If automatic code summarization becomes widespread, these errors may become more relevant, especially in critical use cases such as medical or governmental software.

There were also several categories within the taxonomy created by Huidrom and Belz[13] which did not overlap with our taxonomy. We evaluated the maximally merged dataset within the paper. One interesting case that we did not address is the existence of orthogonal error types, these errors aid in structured categorization of errors without blowing up the taxonomy size. We decided to generalize the errors while classifying them for this paper due to our interest in the broad categories under which LLM categories can be classified but this is an elegant way to allow for additional classification. In addition, their taxonomy contains an "ERROR IN INPUT" category which describes errors which are caused by an error in the input. We excluded these errors as we believed them to be a misuse of the model in the context of the experiment within this paper. It may be interesting however, to analyze whether a model is able to correct the output regardless of an input error. Additionally, Huidrom and Belz included an "Other" category which technically causes full overlap with our taxonomy.

6.4 Implication

One of the major implications which result from Table 1 is the prevalence of LLM specific errors and Semantic errors in this experiment. It seems as if there are still many mistakes which might be corrected with further development. Hallucinations, code snippets, and repetitions are especially common within the error analysis. These errors are all critical due to the way they reduce the quality of a comment significantly by impacting the semantic meaning of a comment.

It also seems as if there are few errors related to linguistics and syntax, meaning that these categories already perform well on average. Additionally, the errors mostly occurred in addition to errors within the Semantic or LLM Specific categories.

Besides error categories, we also found that BLEU-1 and ROUGEL-F1 scores are generally unreliable for accuracy evaluation within the context of multilingual code summation if the reference sequences are made up of public repositories. The scores generally predict accuracy well if they have a similarity score over 0.8 but if they are under that value they do not necessarily reflect the accuracy of an inferred comment.

In addition, we found that the taxonomy that was created in this experiment had significant overlap with similar taxonomies while providing categories that classify LLM specific behavior as well. However, the method by which this taxonomy was created left semantic gaps within the taxonomy such as "Toxicity" and "User safety", in addition to in-depth linguistic analysis.

6.5 Recommendations

We recommend additional attention to repetition errors and late terminations when creating code LLMs as these errors are especially present within the error set that we found might be solvable. The other major errors we found were in-context hallucinations but as these errors can be a wide range of semantic issues that finding can not directly be acted upon. We also found that models have the tendency to predict code snippets when given a file even if a span mask is specifically applied to comment syntax.

Besides this, we do not think that BLEU-1 and ROUGEL-F1 are good metrics to evaluate the similarity of code to inferred documentation and we recommend research regarding other metrics.

We also believe that an analysis of errors that models make helps with the understanding of model behavior, therefore we advocate for increased usage of the open coding methodology in the context of LLMs to aid our understanding.

7 FUTURE WORK

7.1 Other languages

This paper was limited by the languages that were analyzed. It might be interesting to do a similar experiment with other languages, especially a greater data analysis that takes several languages from all major linguistic categories.

7.2 Linguistic analysis

We did not do extensive linguistic analysis as we are not linguists. This means that there may be linguistic peculiarities in the dataset that are interesting but have not been seen or elaborated upon. The analysis was performed at the level of a proficient programmer who is fluent in both Dutch and English, as this is the level of our research team. This might be interesting in combination with the previous point, as there might be a correlation between the performance of languages and the linguistic distance compared to languages that were used to train LLMs.

7.3 Custom tokenizers

We noticed that the performance of models was dependent on the way in which data was tokenized. Existing research identifies improved performance in specialized tasks if the tokenizers are trained with a bias for those subsets[6]. It might be interesting to look at the possibilities of improving performances in different languages by either training custom tokenizers or transliterating script to fit other language systems.

8 LIMITATIONS

We did not address high level code documentation or lines of code in this paper, the behavior of LLMs in this context might be an interesting field to pursue.

Other programming languages. We were limited to Java code, which is a widely used language, but as every programming language is different the errors that occur in a summarization context might also differ.

Limited max token size. The maximum token size was limited to 138, which resulted in a 95% inclusion rate for the original comments. This was mostly done due to performance reasons, as longer predictions naturally take longer to run. We believe that this did not impact the research greatly, as we included a large part of the original dataset and long comments are generally rare. However, it is still recommended to lengthen this maximum in future research if the available computing power allows for it.

Limited model size. Due to computation constraints, we only used models with 7 billion parameters. As LLMs seem to consistently perform better with a higher parameter count, it is better to use a larger model[14]. It might be possible that the distribution of errors is different when more parameters are introduced.

Limited model count. Due to time constraints, we were limited to the analysis of only one model. This is regrettable as there might be a bias for model behavior errors that are specific to CodeQwen1.5; therefore, it would be preferable to perform this experiment with multiple models in the future.

Limited dataset. We were limited in the quality of our dataset as we automatically found Dutch code. The quality was deemed sufficient as it reflects the actual quality of comments that result from human behavior. It might be interesting to perform a similar experiment with a curated dataset which only contains high quality code.

Limited quantitative analysis. We were only able to perform BLEU-1 and ROUGEL-F1 analyses on the dataset due to time constraints. It would have been interesting to perform different metrics and look for score correlations between error types and metric values.

9 CONCLUSION

The goal of this paper was to understand the mistakes make by LLMs when performing code summarization in Dutch. The error taxonomy which was created as a result of the open coding analysis of errors resulted in four broad error categories which can be used to classify mistakes. The existence of these categories partially answers **RQ1** and are as follows:

- Linguistic errors
- Semantic errors
- Model behavior errors
- Syntax errors

The most frequent errors made by the model were semantic in nature or had to do with mistakes which were inherent to LLMs such as repetition or hallucination.

Additionally we performed quantitative analysis of the dataset in regards to BLEU-1 and ROUGEL-F1 scores which revealed that both scores were usable to identify similarity which oftentimes ensured some accuracy if the similarity was high enough. The fact remains however that this method is not ironclad due to its inability to predict partial correctness or cases where the similarity to the original comment is low while the accuracy in the context of code summarization is high. Therefore we question the reliability of these scores as evaluation metrics in this context, which answers **RQ2**.

Finally, regarding the similarity to other error taxonomies within adjacent field asked about in **RQ3**. We found that our taxonomy mostly existed as a super-set of the other taxonomies while sacrificing specificity. This illustrates the cross-disciplinary nature of the taxonomy and indicates the relation between the created taxonomy and its underlying ground truth. Our main take away is the lack of toxicity and safety analysis in out current taxonomy which should be expanded upon if the taxonomy is used to classify errors in a context where these classifications are relevant or desired.

10 RESPONSIBLE RESEARCH

In this section, we will briefly elaborate upon our adherence to the Netherlands Code of Conduct for research integrity[3]. This will be done by explaining how we adhered to the five fundamental principles and by addressing any concerns we may still have.

10.1 Honesty

We have tried to evaluate our findings as extensively as possible and to give as many views of the research as possible. Additionally, we have tried to make clear inclusion and exclusion criteria at every step of the way to ensure our own honesty and adherence to the scientific method.

10.2 Scrupulousness

The methods which were used in this paper all have their origins in existing research or resulted from analysis of the data used during the research. Additionally, any decisions which were made have be clarified to the best of our ability.

10.3 Transparency

We have tried to maximize our transparency by describing every step in our research process. Additionally, all code has been made available publicly, so the experiment can be repeated independently if needed. In addition, the method by which we acquired our dataset has been described, and the dataset itself was published along with this paper.

10.4 Independence

We have maintained neutrality at every step of the way in this research paper, and any possible biases such as choices of language or model have been addressed as limitations of the paper.

10.5 Responsibility

We believe that research into the multilingual availability of LLMs is both scientifically and socially relevant due to the increasing prevalence of this technology in day-to-day life. The field of multilingual availability of this technology has the benefit of increasing accessibility to cutting-edge technology. Even if this paper touches a small section of the wider field, we believe it to be a step in the right direction regarding the ethical use of AI.

10.6 Data

The original dataset consists of open-source Github code which was made public by the original creators. We did not run or use the code in any setting, limiting our usage to analysis as described in the paper. While the practice of mining open source code is frequently used in the field of research, concerns have been raised regarding the usage of these datasets and whether there may be violations of the licenses of the code bases[15]. We do not use any code for the training of LLMs, which is the main problem with the violation of these licences however. We were unable to find a way to mitigate these concerns during this research but chose to include the discussion to highlight our concerns regarding the topic.

10.7 Label transparency

We were limited in the open coding process by our lack of language experts, we were unable to implement extensive safeguards regarding the correctness of our labels. Additionally we were unable to verify the labels using a second reviewer. We have tried to mitigate this concern as much as possible by defining inclusion and exclusion criteria as extensively as possible to create transparency regarding our labeling choices.

10.8 Experiment transparency

The entire process is described extensively, and tools are available to find similar results. One possible limitation is the high computing requirements needed to repeat this experiment, as LLMs of this size require significant computing power to run. Sadly, this fact cannot be mitigated, which creates a limitation on the repeatability of this experiment for people or institutions with limited resources.

ACKNOWLEDGMENTS

I would like to thank my fellow bachelor students who performed this research with me, this paper would not be possible without them.

- Maksym Ziemlewski
- Paris Loizides
- Yongcheng Huang
- Sander Vermeulen

REFERENCES

[1] Toufique Ahmed and Premkumar Devanbu. 2023. Few-shot training LLMs for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (<conf-loc>, <city>Rochester</city>, <state>MI</state>, <country>USA</country>, </conf-loc>) *(ASE '22)*. Association for Computing Machinery, New York, NY, USA, Article 177, 5 pages. https://doi.org/10.1145/3551349.3559555

[2] Toufique Ahmed, Kunal Suresh Pai, Premkumar Devanbu, and Earl Barr. 2024. Automatic Semantic Augmentation of Language Model Prompts (for Code Summarization). In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (<conf-loc>, <city>Lisbon</city>, <country>Portugal</country>, </conf-loc>) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 220, 13 pages. https://doi.org/10.1145/3597503.3639183

[3] KA Algra, LM Bouter, AM Hol, Jan Van Kreveld, Daan Andriessen, Catrien Bijleveld, Roberta D'Alessandro, Jenny Dankelman, and Peter Werkhoven. 2018. Netherlands Code of Conduct for Research Integrity.

[4] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255* (2022).

[5] CodeGemma Team, Ale Jakse Hartman, Andrea Hu, Christopher A. Choquette-Choo, Heri Zhao, Jane Fine, Jeffrey Hui, Jingyue Shen, Joe Kelley, Joshua Howland, Kshitij Bansal, Luke Vilnis, Mateo Wirth, Nam Nguyen, Paul Michel, Peter Choy, Pratik Joshi, Ravin Kumar, Sarmad Hashmi, Shubham Agrawal, Siqi Zuo, Tris Warkentin, and Zhitao et al. Gong. 2024. CodeGemma: Open Code Models Based on Gemma. (2024). https://goo.gle/codegemma

[6] Gautier Dagan, Gabriel Synnaeve, and Baptiste Rozière. ICML 2024. Getting the most out of your tokenizer for pre-training and domain adaptation.

[7] Sergio Cozzetti B. de Souza, Nicolas Anquetil, and Káthia M. de Oliveira. 2005. 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* (Coventry, United Kingdom) *(SIGDOC '05)*. Association for Computing Machinery, New York, NY, USA, 68–75. https://doi.org/10.1145/1085313.1085331

[8] Delft High Performance Computing Centre (DHPC). 2024. DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2.

[9] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. BOLD: Dataset and Metrics for Measuring Biases in Open-Ended Language Generation. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (Virtual Event, Canada) *(FAccT '21)*. Association for Computing Machinery, New York, NY, USA, 862–872. https://doi.org/10.1145/3442188.3445924

[10] Andrew Forward and Timothy C. Lethbridge. 2002. The relevance of software documentation, tools and technologies: a survey. In *Proceedings of the 2002 ACM Symposium on Document Engineering* (McLean, Virginia, USA) *(DocEng '02)*. Association for Computing Machinery, New York, NY, USA, 26–33. https://doi.org/10.1145/585058.585065

[11] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and fairness in large language models: A survey. *Computational Linguistics* (2024), 1–79.

[12] Wayne Harbert. 2006. *The Germanic Languages.* Cambridge University Press.

[13] Rudali Huidrom and Anya Belz. 2022. A Survey of Recent Error Annotation Schemes for Automatically Generated Text. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, Antoine Bosselut, Khyathi Chandu, Kaustubh Dhole, Varun Gangal, Sebastian Gehrmann, Yacine Jernite, Jekaterina Novikova, and Laura Perez-Beltrachini (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid), 383–398. https://doi.org/10.18653/v1/2022.gem-1.33

[14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020.

Scaling Laws for Neural Language Models. arXiv:2001.08361 [cs.LG]

[15] Jonathan Katzy, Razvan Popescu, Arie Van Deursen, and Maliheh Izadi. 2024. An Exploratory Investigation into Code License Infringements in Large Language Model Training Datasets. In *Proceedings of the 2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering* (<conf-loc>, <city>Lisbon</city>, <country>Portugal</country>, </conf-loc>) *(FORGE '24)*. Association for Computing Machinery, New York, NY, USA, 74–85. https://doi.org/10.1145/3650105.3652298

[16] Junaed Younus Khan and Gias Uddin. 2023. Automatic Code Documentation Generation Using GPT-3. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (<conf-loc>, <city>Rochester</city>, <state>MI</state>, <country>USA</country>, </conf-loc>) *(ASE '22)*. Association for Computing Machinery, New York, NY, USA, Article 174, 6 pages. https://doi.org/10.1145/3551349.3559548

[17] Wen Lai, Mohsen Mesgar, and Alexander Fraser. 2024. LLMs Beyond English: Scaling the Multilingual Capability of LLMs with Cross-Lingual Feedback.

[18] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004*, 10.

[19] Mario Linares-Vásquez, Gabriele Bavota, Michele Tufano, Kevin Moran, Massimiliano Di Penta, Christopher Vendome, Carlos Bernal-Cárdenas, and Denys Poshyvanyk. 2017. Enabling mutation testing for android apps. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. 233–244.

[20] Chaoqun Liu, Wenxuan Zhang, Yiran Zhao, Anh Tuan Luu, and Lidong Bing. 2024. Is Translation All You Need? A Study on Solving Multilingual Tasks with Large Language Models. arXiv:2403.10258 [cs.CL]

[21] Junayed Mahmud, Fahim Faisal, Raihan Islam Arnob, Antonios Anastasopoulos, and Kevin Moran. 2021. Code to Comment Translation: A Comparative Study on Model Effectiveness & Errors. In *Proceedings of the 1st Workshop on Natural Language Processing for Programming (NLP4Prog 2021)*, Royi Lachmy, Ziyu Yao, Greg Durrett, Milos Gligoric, Junyi Jessy Li, Ray Mooney, Graham Neubig, Yu Su, Huan Sun, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 1–16. https://doi.org/10.18653/v1/2021.nlp4prog-1.1

[22] Valerie R Mariana. 2014. *The Multidimensional Quality Metric (MQM) framework: A new framework for translation quality assessment*. Brigham Young University.

[23] Paul W. McBurney and Collin McMillan. 2014. Automatic documentation generation via source code summarization of method context. In *Proceedings of the 22nd International Conference on Program Comprehension* (Hyderabad, India) *(ICPC 2014)*. Association for Computing Machinery, New York, NY, USA, 279–290. https://doi.org/10.1145/2597008.2597149

[24] Matthew B Miles and A Michael Huberman. 1994. *Qualitative data analysis: An expanded sourcebook*. sage.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. (10 2002). https://doi.org/10.3115/1073083.1073135

[26] Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How Multilingual is Multilingual BERT?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 4996–5001. https://doi.org/10.18653/v1/P19-1493

[27] Robert Shafer. 1955. Classification of the Sino-Tibetan languages. *Word* 11, 1 (1955), 94–111.

[28] Khetam Al Sharou and Lucia Specia. 2022. A Taxonomy and Study of Critical Errors in Machine Translation. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, Helena Moniz, Lieve Macken, Andrew Rufener, Loïc Barrault, Marta R. Costa-jussà, Christophe Declercq, Maarit Koponen, Ellie Kemp, Spyridon Pilos, Mikel L. Forcada, Carolina Scarton, Joachim Van den Bogaert, Joke Daems, Arda Tezcan, Bram Vanroy, and Margot Fonteyne (Eds.). European Association for Machine Translation, Ghent, Belgium, 171–180. https://aclanthology.org/2022.eamt-1.20

[29] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, Dipanjan Das, and Jason Wei. 2023. Language models are multilingual chain-of-thought reasoners. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=fR3wGCk-IXp

[30] Qwen Team. 2024. Code with CodeQwen1.5. https://qwenlm.github.io/blog/codeqwen1.5/

[31] Arda Tezcan, Véronique Hoste, and Lieve Macken. 2017. SCATE taxonomy and corpus of machine translation errors. *Trends in E-tools and resources for translators and interpreters* (2017), 219–244.

[32] Lieke Verheijen and Roeland van Hout. 2022. Manifold code-mixing in computer-mediated communication: The use of English in Dutch youths' informal online writing. *Ampersand* 9 (2022), 100091. https://doi.org/10.1016/j.amper.2022.100091

[33] Chaozheng Wang, Zongjie Li, Cuiyun Gao, Wenxuan Wang, Ting Peng, Hailiang Huang, Yuetang Deng, Shuai Wang, and Michael R. Lyu. 2024. Exploring Multi-Lingual Bias of Large Code Models in Code Generation. arXiv:2404.19368 [cs.SE]

[34] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[35] Fei Yuan, Shuai Yuan, Zhiyong Wu, and Lei Li. 2024. How Vocabulary Sharing Facilitates Multilingualism in LLaMA? arXiv:2311.09071 [cs.CL]

[36] Xiang Zhang, Senyu Li, Bradley Hauer, Ning Shi, and Grzegorz Kondrak. 2023. Don't Trust ChatGPT when your Question is not in English: A Study of Multilingual Abilities and Types of LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, Singapore, 7915–7927. https://doi.org/10.18653/v1/2023.emnlp-main.491

[37] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL]

[38] Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024. Multilingual Machine Translation with Large Language Models: Empirical Results and Analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 2765–2781. https://aclanthology.org/2024.findings-naacl.176