Research Master Thesis

LLMs as annotators for machine translation quality estimation

Sidi Wang

a thesis submitted in partial fulfilment of the requirements for the degree of

MA Linguistics

(Human Language Technology)

Vrije Universiteit Amsterdam

Computational Lexicology and Terminology Lab
Department of Language and Communication
Faculty of Humanities

Supervised by:	Sophie Arnoult
2nd reader:	Piek Vossen
Submitted:	June 30, 2024

Abstract

Multidimensional quality metric (MQM) is a flexible annotation framework for translation evaluation. The rich information from MQM annotations has been utilized for training machine translation (MT) quality estimation (QE) models, aiming at obtaining high correlations with human judgment data. However, the annotation process often requires using expert human annotators, and thus is time-consuming and expensive.

As large language models (LLMs) with the multi-head attention mechanism demonstrate excellent performance on various benchmark NLP tasks, researchers start to explore LLMs' capabilities in performing MTQE annotation.

In this project, we explore LLMs' behaviors when prompted to generate MQM annotations for the Chinese-to-English language pair and enhance the annotation quality using prompt engineering. We use prompt patterns in prompt design to improve its structural component and conduct four prompting experiments to develop a prompting technique PPbMQM (prompt-pattern-based-MQM) for the MQM annotation task.

We use the annotations generated by PPbMQM to train a score prediction QE model. This model achieves a higher Pearson correlation than the baseline model that was trained on human annotations. Results of the prompting experiments and QE model experiments demonstrate LLMs' capabilities for the MQM annotation task.

Declaration of Authorship

I, Sidi Wang, declare that this thesis, titled *LLMs as annotators for machine translation quality estimation* and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: June 30th 2024

Signed:

Sidi Wang

Acknowledgments

I would like to express my gratitude to my supervisor, Sophie Arnoult, for her unwavering support and guidance throughout this journey.

I am also deeply thankful to Amir and David at TAUS for their expert advice in the QE domain.

A special thanks to my friends, M and K, for their endless encouragement and for always being there to listen, despite being in three different time zones.

I dedicate this thesis to my parents, who have always provided me with unconditional love and support, and to myself, as a mark of a new beginning in my life.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

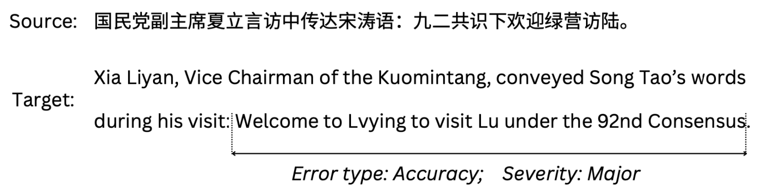
LLMs such as GPT models exhibit human-level performance on various benchmark NLP tasks (Achiam et al., 2023). These LLMs are trained with a vast amount of multi-lingual data enabling their multilingual capability. Recently, many studies have explored using LLMs for cheaper yet reliable annotations for Machine translation (MT) quality estimation (QE).

Quality estimation

The goal of the quality estimation task in the field of machine translation is to automatically assess the quality of translation outputs without depending on reference (human) translations (Specia and Shah, 2018). Usually, the granularity for QE tasks is at the word-level, sentence-level, or system-level, taking human judgment as the gold standard. Given the source and its translation (target) from an MT system as input, word-level QE predicts binary quality labels for the target sentence; sentence-level QE predicts a single quality score (Rei et al., 2022). The system-level QE task produces the rankings of translation systems in (Kocmi et al., 2021).

Multidimensional quality metric (MQM)

A segment in a QE task refers to a source sentence and its target translation produced by an MT system. The ground truth for a segment can be derived from its MQM annotation. MQM is a flexible framework developed by the EU-funded QTLaunch-Pad project. It provides a hierarchy of translation errors with a shared vocabulary (Burchardt, 2013; Lommel et al., 2014).



Source: 国民党副主席夏立言访中传达宋涛语：九二共识下欢迎绿营访陆。

Target: Xia Liyan, Vice Chairman of the Kuomintang, conveyed Song Tao’s words during his visit: Welcome to Lvying to visit Lu under the 92nd Consensus.

Error type: Accuracy; *Severity: Major*

Figure 1.1: An example of MQM

Figure 1.1 is an example of MQM annotation for a Chinese-to-English segment. It contains: 1) a source sentence; 2) a target translation; 3) error(s) details including the category, error spans that are usually indicated by the start and end indexes, and error severity level, which is Major in the example.

Despite the rich information provided by MQM annotations, most automatic metrics only use the final quality score that is calculated based on the number of errors and their severity labels, discarding the error span and error type information (Fernandes et al., 2023; Blain et al., 2023).

Large Language Models (LLMs) for QE annotations

Kocmi and Federmann (2023b) first proposed a prompting technique GEMBA for QE, which is a GPT-based metric that demonstrates the state-of-the-art capabilities at the system-level QE. After that, several prompting techniques like AUTOMQM (Fernandes et al., 2023) and GEMBA-MQM (Kocmi and Federmann, 2023a) as evaluation metrics were developed for QE tasks. However, at the segment level, LLMs do not measure up to the best learned metrics, which are transformer-based models fine-tuned on human judgment data (Fernandes et al., 2023).

1.1 Aim and relevance

In White et al. (2023), prompts are defined as: *"instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output"*. Prompt patterns, on the other hand, *"are a knowledge method analogous to software patterns since they provide reusable solutions to common problems faced in particular contexts"*. Applying prompt patterns in prompt experiments can improve the re-usability by structuring the prompt, providing methodological grounding and eventually, improving the outputs of LLM conversation towards desired directions (White et al., 2023).

Despite showing LLMs' capabilities in the QE task, the above-mentioned prompting techniques fail to follow a structural approach or provide explanations for prompt design. Furthermore, these studies do not conduct a comprehensive analysis of LLMs' behaviors in different dimensions (such as error type, severity, and hallucination analysis). Last but not least, since it is expensive and slow to run the QE task with a system like GPT-4 (Rei et al., 2023), it is worthwhile to investigate using LLM annotations to train smaller-size models that are specifically built for QE tasks. By having such a customized QE model in production, the cost can be reduced compared to directly implementing an LLM of GPT-4 size in the QE pipeline.

In our project, we will explore LLMs' behaviors in generating MQM annotations for the Chinese-to-English language pair. By applying prompting patterns in prompt design and conducting four iterative prompting experiments, we aim to improve the annotation quality at the segment level. One step further, we will investigate what influences LLM-generated MQM annotations will bring to the downstream QE model.

1.2 Problem definition

The main question of this project is: how good are LLMs as annotators for MTQE? Specifically, can we use MQM annotations produced by LLMs for downstream segment-level QE model training?

1.2.1 Sub-question 1

In order to train a QE model that can produce a quality score for each translation segment, we will need high-quality MQM annotations. Our first sub-question is: **how can we enhance the quality of annotations from LLMs at the segment level?**. We will explore this question by designing several prompting experiments and improve the prompt iteratively based on the findings from previous prompting experiments.

1.2.2 Sub-question 2

After we have obtained LLM-generated MQM annotations, to know what influences these annotations will bring to the downstream QE model, we need to answer the second sub-question: **how well does the QE model trained on LLM-generated annotations perform compared to the baseline model trained on human annotations?**

1.3 Contribution

The main contributions of our project include:

- By applying prompting techniques and an iterative prompt development approach, we make the prompting experiments more transparent with a stronger methodological grounding than previous research. We develop a GPT-4o-based prompting technique PPbMQM (prompt-pattern-based-MQM).
- We adapt previous research on MQM evaluation and design an evaluation pipeline that can comprehensively evaluate different aspects of the MQM annotation task. The pipeline includes output parsing, error number/ error type/severity label/error span analysis, and quality score correlation. We have conducted thorough evaluations on LLM-generated annotations using this pipeline.
- We show that the annotations generated by PPbMQM can be used for downstream QE model training and achieve similar or even better performance than the model trained on human annotations. Running QE tasks on such a downstream QE model is cheaper and faster than directly prompting LLMs such as GPT-4 to assess the translation quality.

1.4 Outline

In Chapter 2, we introduce the background of machine translation evaluation and prompt engineering and highlight current studies on prompting for QE annotations.

In Chapter 3, we first present our methodology by introducing the experiment design and the basic prompt design based on prompt patterns. Then we describe the data we used for prompt development and QE model training. At last, we describe the intrinsic evaluation method for prompting experiments, and the QE model training experiment as the extrinsic evaluation method.

In Chapter 4, we describe four prompting sub-experiments for the development of PPbMQM. We analyze the results of each prompting experiment, based on which we improve the prompt for the following prompting experiment.

In Chapter 5, we describe the data annotation, training, and testing for the baseline QE model and the PPbMQM QE model. We analyze the experiment result and conduct a bucket analysis on segments of different qualities on both models.

In Chapter 6 we first discuss the issue revolving around the quality of human annotations. Then we discuss reproducibility from different aspects of prompting experiments: the stability of LLMs, experimental setup documentation, and the debate over proprietary vs. open-source LLMs. Then we discuss the limitations and the ethical issues. Finally, we provide directions for future research.

Chapter 2

Background and Related Work

In this chapter, we first introduce reference-based evaluation metrics and learned metrics for quality estimation that do not depend on reference translations. Next, we provide a guideline for prompt engineering with prompt patterns we can use for our annotation task. Lastly, we present recent research on MQM prompting techniques and the current research limitations.

2.1 Background: Machine translation evaluation

2.1.1 Reference-based evaluation

To automate the evaluation task, the traditional approach usually compares the lexical similarity between the MT translation output with a reference translation which is deemed correct. The comparison can be word-based, like BLEU (Papineni et al., 2002), or character-based like chrF (Popović, 2015). These lexical-based metrics, however, usually highly depend on word matching and fail to capture the overall meaning and the fluency of translation output.

Embedding-based metrics, on the other hand, measure the similarity between the vector representations of the MT translation output and its reference by utilizing embeddings like BERTscore (Zhang et al., 2019). Compared to lexical-based metrics, embedding-based metrics often show a higher level of correlation with human judgment due to their capabilities to capture deeper semantics in a latent space (Lee et al., 2023).

However, all the above-mentioned metrics require high-quality reference translations which are rarely available. Similar to other language generation tasks, in MT evaluation, usually there is not just one correct translation, but a set of correct translations for a given source sentence (Fernandes et al., 2023). Consequently, reference-based metrics are not always feasible, robust, and reliable.

2.1.2 Quality estimation

Different from the above evaluation metrics that need reference translations, quality estimation systems are built to assess the quality of translation outputs without access to reference translations. Taking human judgment as gold standards, QE tasks can be at word-level, sentence-level or system-level. Xenouleas et al. (2019) propose a BERT-based model SUM-QE for quality estimation that obtains high correlations with human ratings in a set of linguistic criteria for generated text. Sellam et al. (2020) propose a

BERT-based learned metric that can model human judgment for text generation tasks. Rei et al. (2022) develop QE systems built upon COMETKIWI and pre-trained on human judgment data that achieve the best results for all three tasks in the WMT 2022 QE Shared Task. Rei et al. (2023) further develop the COMETKIWI-22 model and rank the first in all multilingual tasks in the WMT 2023 QE Shared Task. The architecture of COMETKIWI will be introduced in Section 2.1.5.

2.1.3 Human judgement data

Direct assessment (DA)

DA, in which annotators are asked to rate translation quality for each segment, is a common way to collect human judgment data for QE systems. In the WMT 2023 QE task, the DA data provided for each source-translation pair is annotated using a scale of 0-100 with 100 indicating a perfect translation and 0 representing a completely incorrect translation (Blain et al., 2023).

However, issues arise when directly asking assessors to provide a single score to represent translation quality and can lead to noisy judgments. For instance, when assessing two translation candidates for the same source sentence, how should they quantify the score differences if one is more natural but less accurate than the other (Freitag et al., 2021).

MQM

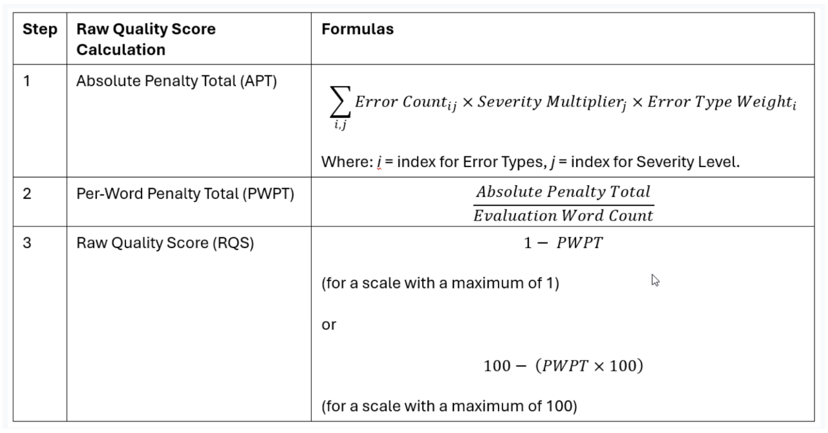
The quality score deriving from the MQM annotations is another common source of human judgment data for sentence-level QE tasks. In the 2023 WMT QE shared task, MQM annotations were used for word-level fine-grained error detection QE systems training, where the error span information was converted to a sequence of Bad/Good labels for all tokens, making use of MQM's rich information (Blain et al., 2023). Compared to DA, MQM provides more detailed information on the identification of errors that is the ground for any scoring and ranking (Freitag et al., 2021).

The current version of MQM typology [1] has eight core error categories: Accuracy, Terminology, Linguistic conventions, Style, Locale conventions, Audience appropriateness, Design and Markup, and Custom, and each top category is further divided into several subcategories. In terms of error severity level, there are Neutral, Minor, Major, and Critical. Based on the needs for different QE tasks, users can customize the error categories and severity levels. This adaptability makes MQM suitable for diverse downstream MT applications.

When converting MQM annotations to quality scores, for weight setting, Freitag et al. (2021) give Minor weight at 1 and Major at 5, the combination of which was found to provide the best balance between stability and ability. When calculating the quality score for each segment, the MQM official website provides "Scoring method without calibration" method [2] (see Figure 2.1). Because of the 5-time weight difference, a major error will be penalized more severely than a minor error. Negative quality score values from this formula will all be mapped to 0. By applying this method to a segment with MQM annotation, a quality score ranging from 0 to 1 can be derived, with 1 indicating perfect quality with no error and 0 bad quality.

[1] `https://themqm.org/the-mqm-typology/`, access date: June 14th, 2024

[2] `https://themqm.org/error-types-2/the-mqm-scoring-models/`, access date: June 7th, 2024



Step	Raw Quality Score Calculation	Formulas
1	Absolute Penalty Total (APT)	$\sum_{i,j} Error\ Count_{ij} \times Severity\ Multiplier_j \times Error\ Type\ Weight_i$ Where: *i* = index for Error Types, *j* = index for Severity Level.
2	Per-Word Penalty Total (PWPT)	$\frac{Absolute\ Penalty\ Total}{Evaluation\ Word\ Count}$
3	Raw Quality Score (RQS)	$1 - PWPT$ (for a scale with a maximum of 1) or $100 - (PWPT \times 100)$ (for a scale with a maximum of 100)

Figure 2.1: Quality score calculation from MQM official website

2.1.4 Metric meta-evaluation

The goal of metric meta-evaluation is to estimate translation from MT systems *"by comparing the agreement between the metric scores and ground-truth quality scores"* (Fernandes et al., 2023), where three correlation scores are usually used for the comparison: Pearson, Spearman, and Kendall's τ coefficient. The Pearson correlation coefficient assumes that two variables are of normal distribution and have a linear relationship. Spearman and Kendall's τ coefficient, on the other hand, computes by rank of two variables that may or may not have a linear relationship. Kendall's τ coefficient calculates the number of concordant and discordant pairs. All three coefficients range from -1 to 1, with 0 indicating two variables are completely independent, 1 if they are the same, and -1 if they are the same but in the opposite direction (Lee et al., 2023).

2.1.5 Learned metrics

LLMs based neural metrics

The multi-head attention mechanism in transformers is the key component that enables a deeper contextualized understanding of input sentences (Vaswani et al., 2017; Liu et al., 2019) and is shown to perform well in many NLP tasks including machine translation. Researchers of automatic evaluation metrics have also started to build MT evaluation frameworks by using transformer-based models, such as XLM-R Large (Conneau et al., 2020)[3] used in Rei et al. (2022). These learned metrics, aiming at assigning a single quality score to a candidate translation, correlate better than traditional metrics like BLEU (Freitag et al., 2022).

The COMET framework

COMET (Cross-lingual optimized metric for evaluation of translation) is a neural framework for *"training highly multilingual and adaptable MT evaluation models that can function as metrics"* (Rei et al., 2020). The COMETKIWI model is built on top of the COMET framework and OPENKIWI (Kepler et al., 2019) and can be trained on human judgment data for QE tasks. Figure 2.2 shows the general architecture of COMETKIWI from Rei et al. (2022).

[3]`https://huggingface.co/xlm-roberta-large`

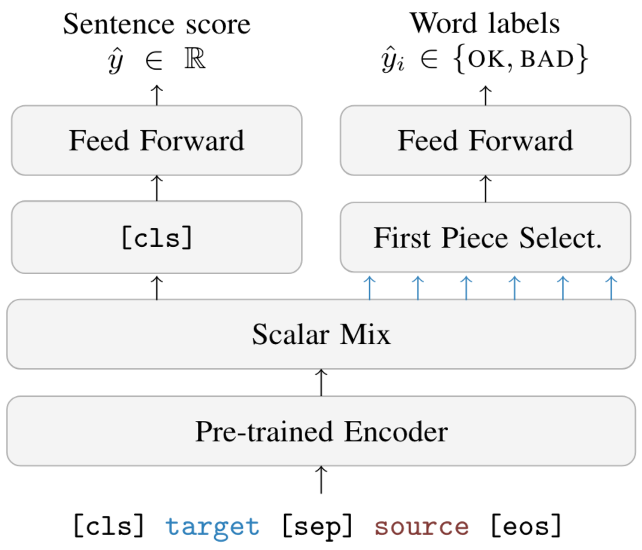


Figure 2.2: General architecture of COMETKIWI

2.2 Background: Prompt engineering

2.2.1 Large language models

GPT-4, with its variants GPT-4.0-Turbo and GPT-4o, is a large-scale transformer-based model pre-trained with the objective of predicting the next token in a document. Its training data includes both publicly available data and licensed proprietary data from third parties (Achiam et al., 2023). Llama 3 is a collection of open-source foundation language models with different scales from 7B to 65B parameters. Unlike GPT-4, LLaMA models are trained completely using only publicly available datasets (Touvron et al., 2023).

Hallucination

Despite LLMs' ever-improving capabilities, they do not always produce truthful responses and have the tendency to hallucinate. Sometimes LLMs generate content that is unfaithful to the input context, or provide false information about the world (Achiam et al., 2023).

2.2.2 Prompt engineering guideline

White et al. (2023) describes a prompt pattern catalog for prompt engineering that can serve as a framework to solve a range of tasks, enabling effective conversations with LLMs. Prompts can be designed using the below patterns from this catalog, combining both deductive and inductive approaches:

The question refinement pattern

1) Intent: this pattern can be used in the prompt engineering process by asking the LLM to refine the input questions, and providing additional information to optimize the prompt.

2) Contextual statement example: *what is the best way to instruct a professional Chinese-to-English translator to assess the quality of a translation?*

The alternative approach pattern

1) Intent: this pattern can mitigate the cognitive biases of the user, enabling the LLM to provide an alternative approach to the task.
2) Contextual statement example: *When you are assessing the translation quality, summarize all potential approaches.*

The reflection pattern

1) Intent: this pattern can to used to obtain LLMs' rationale behind the given answer.
2) Contextual statement example: *Please provide an explanation for each error you identified.*

The cognitive verifier

1) Intent: the pattern is to ask the LLM to convert a high-level question into multiple sub-questions.
2) Contextual statement example: *Please provide other questions that can help to solve the Chinese-English translation quality estimation task.*

The persona pattern

1) Intent: the persona pattern can direct the LLM to take a certain point of view or perspective when generating answers.
2) Contextual statement example: *You are a multilingual Chinese and English speaker and you work as a translation quality annotator.*

The output automater pattern

1) Intent: this pattern can instruct LLM to generate a script or other automation artefact that can reduce the manual editing effort
2) Contextual statement example: *from now on, whenever you generate annotations for each error, provide them in a JSON format with the following keys: error_type, severity, error_span_index, marked_text.*

2.3 Related work: Prompting for MQM style annotations

2.3.1 MQM prompting techniques

Recent works have shown a growing interest in employing LLMs for translation evaluation via zero- and few-shot methods (Fernandes et al., 2023). Prompting techniques, such as GEMBA (Kocmi and Federmann, 2023b), AUTOMQM (Fernandes et al., 2023) and GEMBA-MQM (Kocmi and Federmann, 2023a), demonstrate that LLMs can show state-of-the-art performance at the system-level, but at the segment-level, when compared to learned metrics, their correlations with human judgment data are still low.

Rei et al. (2023) design a five-shot prompt GPT4-QE to predict the location and severity. It performs well in the fine-grained error span detection task. However, sentence-level scores derived from span and severity annotations show a poor correlation with human DA. They also discovered that in the few-shot setting, the number of examples, the order of examples, and the number of errors in each example influence the behaviors of GPT-4 noticeably. Similarly, the GEMBA metric described in Kocmi and Federmann (2023b) also demonstrates that among the four prompt templates tested, the least constrained one has the best performance.

GEMBA and GEMBA-MQM are all GPT-based metrics. However, there are several problems revolving around proprietary LLMs like OpenAI models: firstly, running QE with a system such as GPT-4 is expensive (Rei et al., 2023); secondly, their training processes largely remain opaque in terms of the training data and whether any published test data is included in it; thirdly, the reproducibility of these systems can not be guaranteed since users have no control over its future availability or update (Kocmi and Federmann, 2023a).

Chapter 3

Methodology

In Section 1.1, we concluded the limitations of current MQM prompting techniques and presented our research goal as to improve the quality of segment-level MQM annotations for downstream QE model training.

In this chapter, we describe our approaches in prompting experiment design, data and preprocessing by presenting the experiment design section first, since the design affects our data pre-processing approach. Lastly, we introduce the intrinsic and extrinsic evaluation methods. We derive all quality scores from MQM annotations by applying the formula described in Section 2.1.3.

3.1 Prompting experiment design

Kocmi and Federmann (2023b) mention that the GEMBA prompting technique only works with GPT-3.5 and larger models. Based on this finding, in our project, we select GPT-3.5 Turbo, GPT-4.0 Turbo, GPT-4o and LLaMA 3 70B for prompting experiments.

To explore LLMs' capabilities for the MQM annotation task, we design four prompting experiments. Each sub-experiment is adapted based on the result of the previous experiment. We call this approach iterative prompt development. A prompting technique 'PPbMQM' will be developed based on these four sub-experiments.

3.1.1 Sub-experiment 1: Explore LLMs' knowledge on QE and MQM

In this experiment, we ask LLMs questions in order to test their knowledge of the machine translation quality estimation task and their reasoning capabilities for MQM annotation. All questions are refined using ChatGPT by applying ***the question refinement pattern***. Initially, we will conduct experiments on four models: GPT-3.5, GPT-4 Turbo, GPT-4o, and LLaMA 3.

3.1.2 Sub-experiment 2: Basic prompt and format adjusting

In this second sub-experiment, aiming to examine and adjust the output format, we prompt each LLM with 10 test segments with a basic prompt, the design of which will be explained in Section 3.1.5. Based on the result, the basic prompt is improved and adjusted for the follow-up sub-experiment.

3.1.3 Sub-experiment 3: Basic prompt with 1000 segments

In this sub-experiment, we use 1000 segments to test LLMs' capabilities using zero-shot prompting. The LLM output will be evaluated automatically for error span, error type, severity, quality score correlation, and manually for hallucination.

3.1.4 Sub-experiment 4: Improving basic prompt prompt

Based on the analysis of sub-experiment 3, one LLM will be chosen with a further improved prompt version. Few-shot method will be implemented to improve the performance.

3.1.5 Basic prompt design

Figure 3.1 shows the basic prompt. To reduce the complexity of the task, we will prompt for the top category error types: Accuracy, Fluency, Terminology, Style, and Locale convention with two severity level labels 'Major' and 'Minor'. Three prompt patterns presented in Section 2.2.2 are used for the basic prompt design:

1) The persona pattern: *You are a professional Chinese-English translator.*
This statement asks LLM to act as a specific persona – in this case, a Chinese-English translator for our task, which should evoke traits associated with the persona.

2) The output automater pattern: *Please output a json file with the following keys; Please only generate in json format*
These two statements ask the LLM to provide the output in JSON format, constraining the structure of the output. If the LLM follows the instructions and produces JSON pattern output, the manual editing effort for automatic evaluation can be reduced. Additional instructions also are given for each key in order to obtain output with a standard vocabulary for MQM annotations.

3) The reflection pattern: *5. explanation*
Asking LLMs to produce an explanation for each error identified can help us to understand the reasoning and assumption behind it.

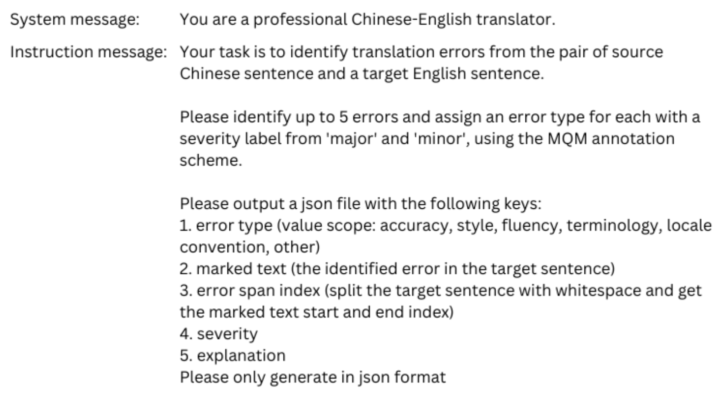
Furthermore, 'using the MQM annotation scheme' is added in order to elicit more domain knowledge that could be possibly acquired during the training phase.

3.1.6 Reproducibility

OpenAI states: *determinism is not guaranteed, and you should refer to the system_fingerprint response parameter to monitor changes in the backend. This fingerprint represents the backend configuration that the model runs with*[1].

In all prompting experiments, we track this value for each request and will conduct analysis by comparing results from different fingerprints. To our knowledge, LLaMA models do not have such a backend system marker value that can be retrieved when calling APIs. The hyperparameter settings for each experiment are well-documented and presented in Appendix A.

[1]`https://cookbook.openai.com/examples/reproducible_outputs_with_the_seed_parameter`, access date: April 24th, 2024.



System message: You are a professional Chinese-English translator.

Instruction message: Your task is to identify translation errors from the pair of source Chinese sentence and a target English sentence.

Please identify up to 5 errors and assign an error type for each with a severity label from 'major' and 'minor', using the MQM annotation scheme.

Please output a json file with the following keys:
1. error type (value scope: accuracy, style, fluency, terminology, locale convention, other)
2. marked text (the identified error in the target sentence)
3. error span index (split the target sentence with whitespace and get the marked text start and end index)
4. severity
5. explanation
Please only generate in json format

Figure 3.1: The basic prompt designed by applying prompt patterns

Column	Description
system	The name of the translation system
doc	The document ID
docSegId	The segment ID in the document
globalSegID	the global segment ID
rater	The rater ID
source	The source Chinese sentence.
target	The target English translation with error span marked by delimiters
category	The concatenated top and sub category
severity	The severity level (Major or Minor)

Table 3.1: Main data fields of the Expert-based human evaluation WMT 2023 generalMT dataset

3.2 Data

The datasets used for this project are Expert-based Human Evaluations for 16 submissions of WMT generalMT 2023 and generalMT 2022 for the Chinese-to-English language pair [2]. The 2023 dataset contains data from news, e-commerce user reviews, and manuals domains (Kocmi et al., 2023). The 2022 dataset contains data from conversation, e-commerce, news, and social media domains (Kocmi et al., 2022). The MQM framework-based annotation methodology is explained in Freitag et al. (2021).

3.2.1 Data description

The 2023 dataset contains 55216 instances and the 2022 data contains 46419 instances in total, with each row representing an error for a segment. Table 3.1 describes the main data fields in the 2023 dataset, with which the 2022 data shares similar field properties.

[2]Github repository:
`https://github.com/google/wmt-mqm-human-evaluation/`

Error type

The error hierarchy includes the five top-level categories: Accuracy, Fluency, Terminology, Style, and Locale convention, each with several sub-categories. The original data uses a top-category ‘Source-issue’ to indicate errors in the source Chinese sentences.

Severity

The error categories include three levels: 1. Major: actual translation and grammatical errors; 2. Minor: smaller imperfections; 3. Neutral: purely subjective opinions about the translation.

Data pre-processing

The goal of data pre-processing is to generate a new dataset with each instance representing a translation pair segment with its MQM annotation and other relevant details, including the system and document/segment ID information. Below are the detailed pre-processing steps for the 2023 dataset using **Pandas** package[3]:

1. Get the **marked text** in the target/source sentence indicated by delimiters. A marked span in the source sentence usually indicates an omission error and most source sentences do not have a marked error span.

2. Get the **marked span index** in source/target sentences by the positions of delimiters. This is done by splitting the sentences by **NLTK** tokenizer for English target sentences and **Jieba** cut method for Chinese source sentences with omission error.

3. Remove delimiters in source and target sentences.

4. Get only the **top category** from the field of category for each error, keeping only the subcategory Omission, which is under the top-level Accuracy. We keep this sub-category to explore how LLMs handle this type of translation error.

5. Concatenate the **top category**, **severity**, **marked text** and **error span index** as the **MQM annotation** for each identified error into a new column

6. Remove rows in which the MQM annotations contain unclear information (such as ‘HOTW’, ‘nan’ and ‘Other’ in the error category).

7. Remove rows with severity labels ‘Neutral’ which indicates only the subjective opinions of annotators, keeping only the ‘Major’ and ‘Minor’ labels.

8. Remove rows with the error category ‘Source-issue’, since our project focuses on identifying errors in the target translation.

9. Group the DataFrame by fields of system, doc, docSegID, globalSegID, rater, source, and target to get a new DataFrame with each row representing a segment with its MQM annotation

[3]We use a similar approach to process the 2022 dataset

10. Remove segments with more than 5 errors.[4]
11. Remove outlier segments where the target sentences are shorter than 2 tokens or longer than 100 tokens.
12. Randomize the DataFrame[5] and reset index.

After pre-processing, we obtained 13970 segments with MQM annotations from the 2023 dataset and 29344 segments from the 2022 dataset. We further split these segments into three datasets for different experiments:

- A prompt development dataset with 1000 segments from the 2023 dataset
- A QE model train dataset with 20703 segments from the 2022 data (we consider around 20000 segments to be sufficient for QE model training)
- A QE model test dataset with 5000 segments from the 2023 data

3.2.2 How well do human annotators agree with each other?

In the 2023 dataset, each annotator annotated different segments, thus the annotation agreement between annotators can not be calculated.

In the 2022 dataset, we tested the correlations of quality scores between different annotators for the same segments. From Table 3.2 we can conclude that the agreements between annotators are very low or even negative.

Annotators	Number of segments	Pearson	Spearman
rater1, rater2	702	0.287	0.274
rater1, rater3	1049	0.036	0.063
rater1, rater6	299	0.059	0.088
rater1, rater5	11	-0.407	-0.337
rater1, rater4	13	-0.143	-0.06
rater2, rater3	515	-0.005	-0.042
rater2, rater6	280	0.128	0.07
rater2, rater4	22	0.212	0.062
rater3, rater6	736	0.043	0.009
rater3, rater8	21	-0.101	0.024
rater4, rater6	16	-0.22	-0.011

Table 3.2: Agreement between annotators in the Expert-based human evaluation of WMT generalMT 2022 dataset (Chinese-to-English)

3.3 Evaluation

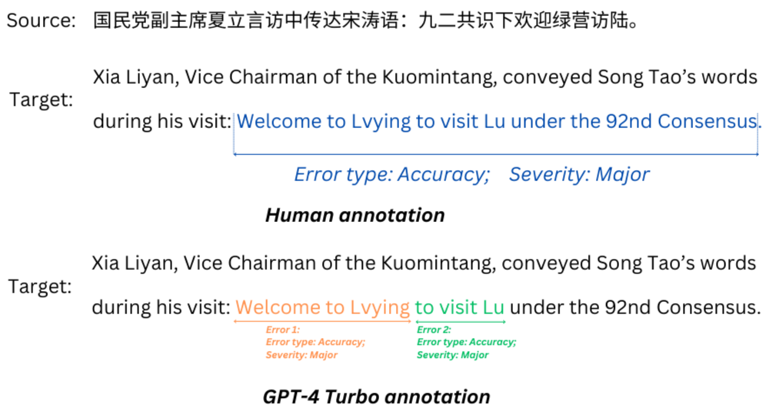
In Section 3.1, we described the design of four prompting experiments. For the first and second experiments, the results will be evaluated manually. For sub-experiments 3 and

[4]In the work of Freitag et al. (2021), they mentioned that they imposed a maximum of 5 errors per segment in the annotator guideline. However, segments with more than 5 errors were found during pro-processing. We decided to remove these segments for the purposes of consistency.

[5]Random state: 31415926

4 in which we test with 1000 segments, we will conduct intrinsic automatic evaluation, and additionally hallucination analysis in sub-experiment 3.

3.3.1 Intrinsic evaluation


Source: 国民党副主席夏立言访中传达宋涛语：九二共识下欢迎绿营访陆。

Target: Xia Liyan, Vice Chairman of the Kuomintang, conveyed Song Tao's words during his visit: Welcome to Lvying to visit Lu under the 92nd Consensus.

Error type: Accuracy; Severity: Major

Human annotation

Target: Xia Liyan, Vice Chairman of the Kuomintang, conveyed Song Tao's words during his visit: Welcome to Lvying to visit Lu under the 92nd Consensus.

Error 1: Error type: Accuracy; Severity: Major

Error 2: Error type: Accuracy; Severity: Major

GPT-4 Turbo annotation

Figure 3.2: A segment with human and GPT-4 Turbo MQM annotations

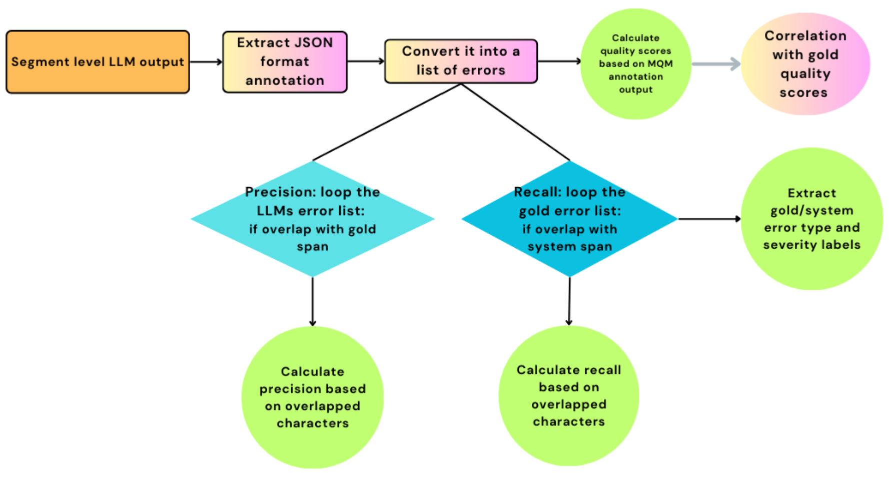

Segment level LLM output
Extract JSON format annotation
Convert it into a list of errors
Calculate quality scores based on MQM annotation output
Correlation with gold quality scores
Precision: loop the LLMs error list: if overlap with gold span
Recall: loop the gold error list: if overlap with system span
Extract gold/system error type and severity labels
Calculate precision based on overlapped characters
Calculate recall based on overlapped characters

Figure 3.3: Flowchart: intrinsic evaluation

Multi-level and multi-dimensional evaluations will be conducted at the segment level (Figure 3.2 shows an example of a segment with human and GPT-4 Turbo annotations): for error span, we will follow the approach implemented by the WMT 2023 QE shared task (Blain et al., 2023) by calculating the recall, precision, and f1; for severity and error type, if the LLM annotated span overlaps with human-annotated span, we will extract the severity and error type labels in order to compare with the human annotations; for quality score evaluation, we will calculate the quality score for each segment, and then calculate the correlation scores between system quality scores and gold quality scores. For each segment, the detailed steps, as demonstrated in Figure 3.3, include:

1. Extract JSON format MQM annotations from LLMs' output.
2. Convert the annotations into a list of errors and calculate the quality scores for metric meta-evaluation obtaining **quality score Pearson and Spearman correlations**.
3. For each error in human annotations, find the best matching LLM outputted error with the highest character overlap by matching the start and end indexes with the gold indexes; Calculate the **error span recall** for each gold error that equals to the ratio of the overlap character length to the gold error length (or 0 if no overlap); Extract the error type and severity labels from the human-annotated error and its best matched LLM annotated error for **severity and error type evaluation**.
4. For each LLM generated error, find the best matching gold error with the highest character overlap by matching the start and end indexes with the gold indexes; Calculate the **error span precision** for each LLM annotated error that equals the ratio of the overlap character size to the LLM outputted error length (or 0 if no overlap).
5. Average recall/precision scores of errors as the final segment recall/precision score for the segment

3.3.2 Extrinsic evaluation

For extrinsic evaluation, we compare the performance of two sentence-level QE models. The first model will serve as a baseline model and be built on the COMET framework. The training input segments are from the QE model training dataset described in Section 3.2.1. The gold quality scores for these segments are derived from human MQM annotations.

We will use the same settings for the second model as for the baseline model except for quality scores, which will be derived from the annotations produced by our prompting technique.

Test set

The test set below will be used to test the above-mentioned two models:

- The QE model testing dataset with 5000 segments that were introduced in Section 3.2.1.

Initially, we also selected a second test set taken from *WMT 2022 QE task 1 mqm* that consisted of 500 segments with gold quality scores (z-score) [6], which would enable us to compare our QE models with other 12 participated systems in this shared task[7].

[6] `https://github.com/WMT-QE-Task/wmt-qe-2022-data/tree/main/test_data-gold_labels/task1_mqm/zh-en`

[7] Results: `https://codalab.lisn.upsaclay.fr/competitions/6866#results`; access data: June 14th, 2024

However, we later discovered that this test set was created using the segments from the WMT generalMT 2022 dataset that was also the source of our QE model training dataset. Then we did a duplicate check for these 500 segments with our QE model training dataset and found that 373 source sentences overlapped. To avoid the issue of data leakage, we decided to discard this test set.

Chapter 4

Prompting for MQM annotations

This chapter describes four sub-experiments conducted using an iterative prompt development approach to investigate the LLMs' behaviors on the MQM annotation task. The analysis of the previous sub-experiment is used to provide direction for the follow-up prompting experiment.

4.1 Sub-experiment 1: Explore LLMs' knowledge of QE and MQM

In this sub-experiment, we explored LLMs' knowledge of machine learning quality estimation and details of API parameters can be found in Appendix A.1. Questions were related to the QE task, which had been refined using ChatGPT by applying ***the question refinement pattern***. For example, the first question was refined by ChatGPT with the initial input "*Please suggest a better version of the question: what is machine translation quality estimation?*".

4.1.1 Results analysis

All responses from four LLMs were assessed by a scoring schema ranging from 1 to 5:

- 1 point: the answer is completely incorrect, irrelevant, and nonsensical. It does not address the question at all.
- 2 points: the answer is partially correct but contains significant errors or is incomplete.
- 3 points: the answer covers the basic information and is mostly correct.
- 4 points: the answer is correct, clear, and detailed.
- 5 points: the answer is comprehensive and shows a strong reasoning capability.

Table 4.1 summarizes points for each question of GPT-3.5, GPT-4 Turbo GPT-4o, and LLaMA 3.

Question	GPT-3.5	GPT-4 Turbo	GPT-4o	LLaMA 3
1. Can you explain what machine translation quality estimation is?	3	4	4	4
2. Could you provide an overview of the Multidimensional Quality Metrics (MQM) annotation scheme in around 150 words?	3	4	4	4
3. What are the core error categories of MQM?	2	3	4	4
4. How can I effectively evaluate the Chinese sentence and its English translation using the MQM annotation scheme? Provide a concise response within 100 words, please.	3	5	5	4
5. How would you, as a language model, annotate the translation of a Chinese source sentence into English using the MQM scheme? Please provide an example	3	4	3	3
Total score	14	20	20	19

Table 4.1: Results of sub-experiment 1: Explore LLMs' knowledge of QE and MQM

1. Can you explain what is machine translation quality estimation?

All four LLMs provide valid answers, yet GPT-3.5 states that quality estimation is "typically done by comparing the machine-translated output with reference translations", which is mostly not the case.

2. Could you provide an overview of the Multidimensional Quality Metrics (MQM) annotation scheme in around 150 words?

All four LLMs listed the below key characteristics of MQM:

- It is an annotation scheme for translation quality estimation;
- It contains multiple category dimensions such as Fluency and Accuracy
- It supports consistent and effective evaluation by standardizing quality assessment.

Besides, LLaMA 3 mentioned that MQM provided a standardized vocabulary and methodology, while GPT 4 turbo and GPT-4o highlighted its flexibility and adaptability for diverse applications. In addition, only LLaMA 3 included that MQM defined severity levels.

3. What are the core error categories of MQM?

On the MQM official website [1] there are currently seven high-level error type dimensions: Terminology, Accuracy, Linguistic conventions (Fluency in version 1), Style, Locale conventions, Audience appropriateness (Verity in version 1), Design and markup.

[1] `https://themqm.org/error-types-2/typology/`; access date: May 9th, 2024

LLaMA 3 listed eight categories: Accuracy, Fluency, Terminology, Locale convention, Style, Verity, Design, and Internationalization.

GPT-3.5 listed seven: Accuracy, Fluency, Terminology, Style, Locale-specific, Consistency errors, and Compliance errors.

GPT-4 Turbo listed six categories: Accuracy, Fluency, Terminology, Locale convention, Design, and Verity.

GPT-4o listed eight categories: Accuracy, Fluency, Terminology, Style, Locale convention, Verity, Design, and Internationalization.

The responses show that all LLMs outputted categories with the previous version category name, such as 'Fluency' for 'Linguistic conventions', and 'Verity' instead of 'Audience appropriateness' from LLaMA 3 and GPT 4.0. This was possibly due to the fact that the training data of LLMs had been cut off before the MQM error categories were updated.

In terms of numbers of correct matches with top categories, GPT-4o and LLaMA outputted the same 8 top categories with a correct match of six, one extra category of 'Internationalization', and one 'Design' instead of 'Design and markup', while GPT-4 Turbo also matched six correctly but missed 'Style'. GPT-3.5 was the least accurate, only with a complete match of four.

4. How can I effectively evaluate the Chinese sentence and its English translation using the MQM annotation scheme? Provide a concise response within 100 words, please.

GPT-3.5 did not provide executable steps, only included identifying error types as steps, nor mention assigning severity level. LLaMA 3, GPT-4 Turbo, and GPT-4o provided comprehensive and executable steps including identifying errors using MQM categories, assigning severity levels, and calculating the quality score. In addition, GPT-4 Turbo and GPT-4o mentioned customizing error categories based on the specific context, which demonstrated MQM's flexibility, one of its core characteristics.

5. How would you, as a language model, annotate the translation of a Chinese source sentence into English using the MQM scheme? Please provide an example.

GPT-3.5, GPT-4o, and LLaMA 3 only include identifying different error types without mentioning assigning severity labels while GPT-4 Turbo provides a more systematic step-by-step approach from error identification, error categorization, and severity assessment to documentation. Aside from that, all LLMs give very short no-error segments as examples, thus no error span was mentioned.

Sub-Conclusion

In this sub-experiment, it can be concluded that GPT-3.5 displayed less knowledge base and reasoning capability for the machine translation quality estimation tasks than the other three LLMs. Based on the results from this experiment, we decided to remove GPT-3.5 for further prompting experiments.

4.2 Sub-experiment 2: Basic prompt and format adjusting

A clean and consistent output format benefits the automatic evaluation. This sub-experiment was conducted to explore how well the LLMs adhere to the prompt instruction with regard to the output format. We tested 10 segments by applying the basic prompt on three LLMs. The detailed LLM setups can be found in Appendix A.2. We kept API parameters the same as those in their official playgrounds[2], since we assumed that the settings had been tested sufficiently and could display stable performances.

4.2.1 Experiment result

Results of this sub-experiment exhibit the following:

1. GPT-4 Turbo, GPT-4o, and LLaMA 3 all provided clear and readable values for fields of required keys following the instruction. The output can be parsed using 'json.load()' function after removing the beginning and the end markers.

2. For all systems, the error span indexes can't be accurately mapped to the indexes obtained from MQM human annotations. During the pre-processing, the NLTK package was used to tokenize target sentences, and the start and end indexes were obtained by the positions of error span delimiters, while LLMs did not seem to utilize the same or even a stable and consistent method when generating indexes for error spans.

For the following experiments, to solve the index mapping issue, an automatic method was devised to obtain the start and end error span indexes for each error by the position of the 'marked text' in the target translation:

- If the 'marked text' occurs only once in the target translation, the final error span will be obtained by searching its position in the target sentence;

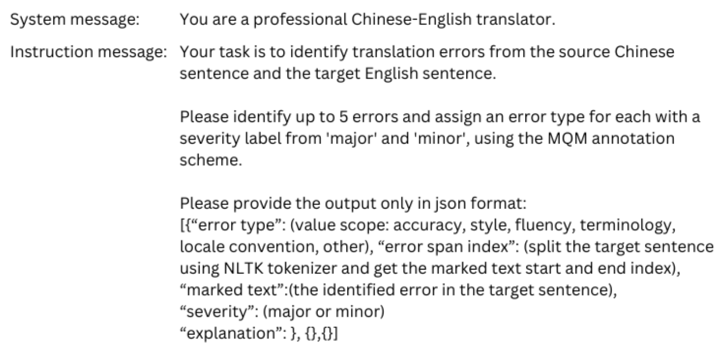
- If the 'marked text' occurs more than once in the target translation, the final error span will be mapped to the nearest one with the start index of the LLM outputted error span index.

For example, if the marked text is 'her father' and it appears twice in the target with starting index 5 and 15 respectively, and the LLM output starting index is 7, the index of 5 will be chosen as the start index for this error because the distance with the LLM output index is smaller.

4.3 Sub-experiment 3: Basic prompt with 1000 segments

From the result of the above sub-experiment 2, we updated the basic prompt by adding 'using NLTK tokenizer' to better direct the LLM to output accurate index values with a clearer JSON format example. Figure 4.1 shows the updated version of the basic prompt:

[2]Llama 3: `https://replicate.com/meta/meta-llama-3-70b-instruct?input=python`; OpenAI: `https://platform.openai.com/playground/`, access date: April 25th, 2024.



System message: You are a professional Chinese-English translator.

Instruction message: Your task is to identify translation errors from the source Chinese sentence and the target English sentence.

Please identify up to 5 errors and assign an error type for each with a severity label from 'major' and 'minor', using the MQM annotation scheme.

Please provide the output only in json format:
[{“error type”: (value scope: accuracy, style, fluency, terminology, locale convention, other), “error span index”: (split the target sentence using NLTK tokenizer and get the marked text start and end index), “marked text”:(the identified error in the target sentence), “severity”: (major or minor)
“explanation”: }, {},{}]

Figure 4.1: Sub-experiment 3: the updated prompt based on the result of sub-experiment 2

4.3.1 Results analysis

We tested with our 1000 instance development dataset respectively on GPT-4 Turbo, GPT-4o and LLaMA 3 (with detailed setup presented in Appendix A.2), and parsed the output for each segment for further evaluation. Table 4.2 shows the results of this sub-experiment.

LLM	Parsable output	Error numbers (Major in parathesis)	Error span f1	Severity - macro avg f1	Error type - macro avg f1	Quality score - Pearson	Quality score - Spearman
GPT-4 Turbo	692	1237 (684)	0.359	0.49	0.24	0.383	0.343
GPT-4o	957	3146 (1256)	0.318	0.51	0.26	0.485	0.382
LLaMA 3	885	2286 (1143)	0.296	0.49	0.16	0.486	0.401

Table 4.2: Sub-experiment 3: results of three LLMs

Output parsing

GPT-4 Turbo: For the 1000 output strings of GPT-4 Turbo, first, we removed the leading and trailing characters at the beginning and at the end. Next, we used *“json.loads()”* method and key values provided in the basic prompt to extract data for each field. As a result, 692 valid Python dictionaries, with each dictionary representing the MQM annotation for a segment, were obtained and evaluated with human annotations.

A deeper look at the invalid output strings revealed that one of the reasons was the presence of invalid key values that were outside the scope instructed in the basic prompt, e.g., *“marked Yue”* instead of the instructed “marked text”.

GPT-4o and **LLaMA 3**, on the other hand, displayed more robustness in producing valid JSON outputs, resulting in 957 and 885 valid output dictionaries.

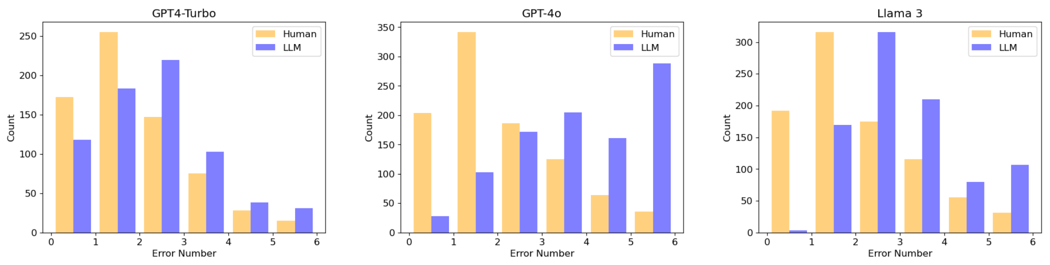


Figure 4.2: Sub-experiment 3: error number distribution with yellow indicating the counts of human identified errors and blue the LLM generated errors

Error number

Figure 4.2 shows the histplots of error number distribution. **GPT-4.0 Turbo:** As the basic prompt instructed, GPT-4 Turbo generated 5 or less than 5 errors for each segment with an average of 1.79 errors, in comparison to 1.39 in human annotations.

In total, 1237 errors were identified by GPT-4 Turbo for these 692 segments whereas 961 errors were identified in human annotations. Furthermore, the system generated all severity (Major/Minor) labels correctly following the instructions of the prompt. However, a handful of incorrect error type labels were generated[3]: *punctuation* (6); *fluentcy* (1); *fluent* (1). These labels were removed in the evaluation.

GPT-4o: In total, 3146 errors were identified for these 957 segments, while human annotations identified 1525 errors. As the basic prompt instructed, the number of errors of each segment was all less than or equal to 5. For each segment, GPT-4o produced 3.29 errors in comparison to 1.59 from human annotations. Just like GPT-4 Turbo, GPT-4o also followed the instructions generating only Major and Minor labels for error severity. However, a small portion of out-of-scope error type labels was also discovered: *punctuation (13), termology (2), consistency (2), redundancy (1), translation (1), translation accuracy (1), clarity (1), term (1)*, which were removed during evaluation.

LLaMA 3: LLaMA 3 identified 2286 errors for these 885 segments resulting in 2.58 errors per segment. Human annotations, on the other hand, had 1389 errors identified in total and an average of 1.57 errors for each. LLaMA 3 followed the value scopes of severity and error type and produced labels as instructed in the basic prompt.

Error severity

Figure 4.3 shows three confusion matrix figures of Severity labels for each LLM.

GPT-4 Turbo: Among all generated errors, 684 were Major and 553 were Minor resulting in a Major-to-Minor ratio of 1.24. However, in human annotations, this ratio was 0.24. That is to say, GPT-4 Turbo tended to identify more Major errors than human annotators. The Major labels had a higher recall score of 0.78 compared to 0.39 for Minor. 225 Minor errors were misclassified as Major while only 30 Major errors were misclassified as Minor.

[3]The value in parentheses indicates the count of the label

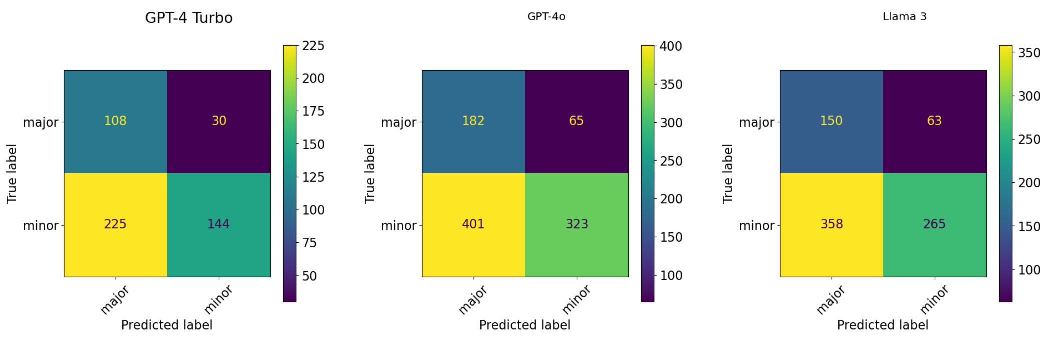
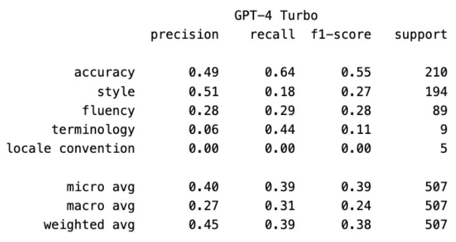


Figure 4.3: Sub-experiment 3: confusion matrix of severity

GPT-4o: GPT-4o generated 1256 Major and 1890 Minor labels, resulting in a ratio of 0.66 compared to 0.27 in human annotations. The Major labels had a recall rate of 0.74 and the Minor 0.45. The confusion matrix showed that for the Minor class, 401 Minor errors were misclassified as Major while only 65 Major errors were misclassified as Minor.

LLaMA 3: For 885 segments, LLaMA 3 generated 1143 Major and 1143 Minor errors (ratio 1.0), while in human annotations there were 306 Major and 1083 Minor errors (ratio 0.28). The Major labels had a recall rate of 0.70 and the Minor 0.43. 358 Minor errors were misclassified as Major, compared to only 63 Major errors that were misidentified as Minor.

Error type



```
                          GPT-4 Turbo
                   precision    recall  f1-score   support

          accuracy       0.49      0.64      0.55       210
             style       0.51      0.18      0.27       194
           fluency       0.28      0.29      0.28        89
       terminology       0.06      0.44      0.11         9
 locale convention       0.00      0.00      0.00         5

         micro avg       0.40      0.39      0.39       507
         macro avg       0.27      0.31      0.24       507
      weighted avg       0.45      0.39      0.38       507
```

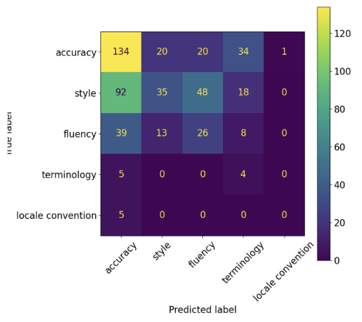
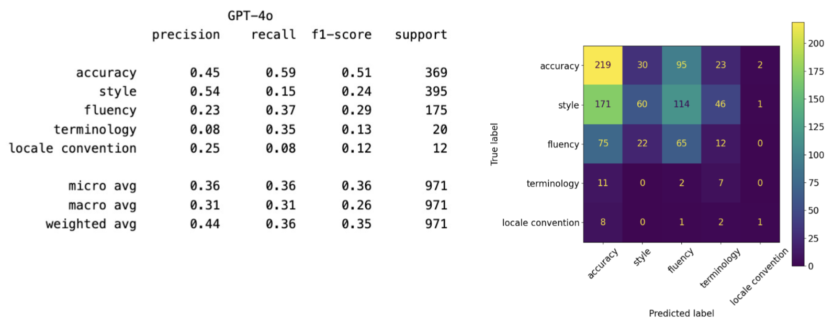


Figure 4.4: Sub-experiment 3: GPT-4 Turbo error type classification report and confusion matrix

Figures 4.4, 4.5 and 4.6 show the error type confusion matrix and classification report for each LLM. Below are the key insights:

- Three classification reports show that Accuracy, Style, and Fluency were the top three common labels. Accuracy had the highest recall and Style the lowest.

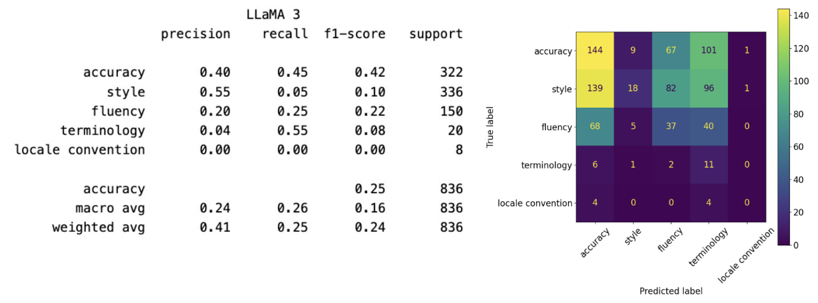


```
                        GPT-4o
              precision    recall  f1-score   support

        accuracy       0.45      0.59      0.51       369
           style       0.54      0.15      0.24       395
         fluency       0.23      0.37      0.29       175
     terminology       0.08      0.35      0.13        20
locale convention      0.25      0.08      0.12        12

       micro avg       0.36      0.36      0.36       971
       macro avg       0.31      0.31      0.26       971
    weighted avg       0.44      0.36      0.35       971
```

Figure 4.5: Sub-experiment 3: GPT-4o error type classification report and confusion matrix



```
                        LLaMA 3
              precision    recall  f1-score   support

        accuracy       0.40      0.45      0.42       322
           style       0.55      0.05      0.10       336
         fluency       0.20      0.25      0.22       150
     terminology       0.04      0.55      0.08        20
locale convention      0.00      0.00      0.00         8

        accuracy                           0.25       836
       macro avg       0.24      0.26      0.16       836
    weighted avg       0.41      0.25      0.24       836
```

Figure 4.6: Sub-experiment 3: LLaMA 3 error type classification report and confusion matrix

- Fluency was often misclassified as Accuracy by three LLMs, additionally, as Terminology by LLaMA 3.

- GPT models' f1 scores achieved a macro average of 0.25, higher than LLaMA-3's score of 0.16

4.3.2 Hallucination

To conduct analysis on the hallucination behaviors of LLMs on our task, 20 extra errors were sampled for each LLM, resulting in 60 MQM error annotations in total. These errors all had zero overlapped span with human annotations. We analyzed these errors by combining the output of the key "explanation", in which we asked LLMs to provide the reasoning and assumptions behind the error identified.

LLMs	Hallucination cases (Major)	Non-hallucination cases (Major)	Uncertain cases (Major)
GPT-4 Turbo	3 (2)	14 (6)	3 (2)
GPT-4o	5 (4)	12 (3)	3 (2)
LLaMA 3	7 (4)	6 (3)	7 (3)

Table 4.3: Hallucination analysis (the numbers in parentheses represent the count of Major errors)

In the non-hallucination cases of these extra errors that LLMs made, common cases were LLMs suggesting more natural and accurate word usages that are more contextually appropriate or idiomatic than the literal translations. The hallucination cases, on the other hand, included making the exact same corrections as the source sentence or identifying the correct translation as erroneous and suggesting the opposite translation that was not aligned with the source.

Due to the ambiguity in the source sentence, some of the uncertain cases were difficult to judge by LLMs. Some cases suggested explaining further an abbreviation occurring in the target sentence "for the audience who may not be familiar with the abbreviation".

Table 4.3 also presents the counts of Major errors in parenthesis. For the cases of hallucination and uncertainty, GPT-4 Turbo, GPT-4o and LLaMA 3 had respectively 4 (0.67), 6 (0.75) and 7 (0.5) Major errors. From this data, it can be concluded that in our sample, percentage-wise, LLaMA 3 identified fewer Major errors than the two GPT-4 models.

Overall, the existence of these non-hallucination cases indicated that LLMs were able to identify the nuanced and subtle word usages that human annotators failed to capture, calling the need for re-examining and refining the human annotations. The hallucination cases revealed that LLMs still had reasoning limitations when performing the task making plausible-sounding but unreasonable answers. Nevertheless, the uncertain cases made us aware of the inherent complexities of our task.

4.3.3 Error span recall, precision and f1

LLM	Error numbers	Recall	Precision	F1
GPT-4 Turbo	1237	0.406	0.42	0.359
GPT-4o	3146	0.422	0.351	0.318
LLaMA 3	2286	0.381	0.377	0.296

Table 4.4: Sub-experiment 3: recall, precision and f1 on avg segment error span

Table 4.4 shows the error span recall, precision, and f1 scores of three LLMs, of which GPT-4 Turbo had the highest precision and f1, while GPT-4o had the highest recall. When combining the factor of the number of total generated errors, it can be found that despite GPT-4 Turbo only identified 1237 errors, its recall was still higher than LLaMA 3, which produced 2286 errors.

4.3.4 Quality score correlation

The Pearson and Spearman correlations were calculated for three LLMs, among which LLaMA 3 scored the highest in both (see Table 4.1). Interestingly, the result was reversed in error span f1 scores: LLaMA 3 had the lowest score of 0.296. In addition, from the hallucination analysis in Section 4.3.2 we found that in our sample LLaMA 3 had the highest number of hallucinations and uncertain cases.

In quality score correlation, for each segment with a certain token length, the factors being taken into account include the number of errors and their severity levels, as Major is penalized 5-times higher than Minor. Thus, to achieve a high correlation, LLMs should be able to label Major and Minor errors with a ratio that is as close as possible to human annotations. In our experiment, LLaMA 3 had the lowest error span f1, which indicated that compared to GPT-4 models, it failed to identify accurate error spans that overlapped with human annotations. However, the Major to Minor ratios in segments may be closer to human judgment, leading to high correlations. This could also possibly provide an explanation for its high tendency toward hallucinations: despite potentially having more hallucination cases, these cases were mostly Minor and penalized less in quality scores, or had a good balance between Major and Minor cases compared to human annotations.

4.3.5 Additional stability test

In this sub-experiment, we added an additional stability test. We modified the basic prompt by removing the part where we requested LLM to generate an explanation for each error while keeping all the other setups the same. We tested on the first 400 segments and compared the result with the original setup for the same segments. Table 4.5 shows the result, demonstrating that GPT-4o showed a stable performance in the stability test, obtaining scores in reasonable ranges with the original.

Test	Parsable output	Error span f1	Severity - macro avg f1	Error type - macro avg f1	Quality score - Pearson	Quality score - Spearman
GPT-4o original test	380	0.307	0.46	0.23	0.452	0.345
GPT-4o stability test	386	0.308	0.49	0.20	0.457	0.376

Table 4.5: Result of the additional stability test with 400 segments

4.3.6 Sub-conclusion

From this sub-experiment, we discovered that three LLMs shared similar behaviors listed below:

- All three LLMs followed the prompt error number restrictions, the errors predicted for each segment are all less than or equal to 5.
- In general, three LLMs tended to predict more errors with higher severity levels compared to human annotators.
- The ‘Style’ error types were the most commonly misclassified category. Despite ’Style’ being among the most frequent error types identified by human annotators, in our experiment, all three LLMs failed to identify most of them with recall scores around 0.15 for two GPT models and only 0.05 for LLaMA 3.

On the other hand, different behaviors were also detected among these three LLMs.

- **Output Parsing:** GPT-4 Turbo produced more problematic JSON outputs that can not be easily parsed and evaluated automatically, while GPT-4o displayed the most robust capability in producing consistent JSON outputs.

- **Error numbers:** GPT-4o identified the highest number of errors (3146) among all compared to LLaMA 3 (2286) and GPT-4 Turbo (961).
- **Error span identification:** GPT models scored approximately 0.05 higher in error span f1 than LLaMA 3, suggesting that they predicted more accurate error spans that were aligned with human annotations (see Table 4.2).
- **Error type:** GPT models demonstrated better capabilities in distinguishing different error types than LLaMA 3.
- **Hallucination:** LLaMA 3 tended to hallucinate more often than GPT models.
- **API stability:** The API of LLaMA 3 provided by Replicate didn't display stability in this sub-experiment. During the requests in the batch that included our 881 segments, its API stopped responding multiple times returning errors like '*ModelError: Prediction interrupted; please retry (code: PA)*' or '*The read operation timed out*'. On the contrary, the OpenAI API ran smoothly without any error, showing more stability than the LLaMA 3 API.

Overall, compared to GPT-4 Turbo and LLaMA 3, GPT-4o produced more automatically parsable JSON format annotations and achieved high error span f1 scores as GPT-4 Turbo. Its macro average f1 scores of severity and error type are also the highest among all LLMs. Despite LLaMA 3 having a 0.01 higher Pearson correlation score and a 0.02 higher Spearman score than GPT-4o, LLaMA 3 tended to display a stronger hallucinating tendency and a less stable API. Cost-wise, GPT-4 Turbo is twice as expensive as GPT-4o. Considering the stability of API, capability of generating parsable JSON format, tendency of hallucination, and cost, GPT-4o was selected for sub-experiment 4.

4.4 Sub-experiment 4

In this sub-experiment, we developed the basic prompt based on the findings from the previous sub-experiments.

4.4.1 Prompt design

What errors did GPT-4o miss?

In total, GPT-4o completely missed 544 errors (with span recall scores of 0). Among these, 75 were Major errors and 479 were Minor errors. For the combinations of error type and severity, *Fluency + Minor*, *Style + Minor*, and *Accuracy + Minor* were the top 3 missing types. In addition, since 'Omission' was not included in the basic prompt, all 77 Omission errors were missed.

4.4.2 Prompt development

The above analysis suggested several directions to enhance the correlation with human annotations.

Firstly, since GPT-4o tends to identify more errors with higher severity than human annotators, we would want to mitigate the severity level and total errors identified. We then devised a scaling approach by converting the severity labels Major and Minor to a scale of 1 to 5, with 1 representing the least severe and 5 most severe. This approach allowed for better control over the total errors identified and their severity levels. This was because based on the severity scale number, errors with lower severity can be removed. In addition, by mapping severity scale numbers to Major or Minor labels, we can balance the number of these two labels.

Secondly, the macro f1 score of error type was only 0.26 indicating that GPT-4o performed poorly in assigning the correct error type labels. We wanted GPT-4o to have a better understanding, so we decided to provide detailed explanations of each error type.

As mentioned in Section 2.3.1, Rei et al. (2023) discovered that the behaviors of five-shot prompt GPT4-QE were influenced greatly by the number of examples included in the prompt, the order of examples, and the number of errors in each example. According to Kocmi and Federmann (2023b), the least constrained prompt template showed the best performance. Ideally, for the annotation task, we want to leverage LLM's multilingual capability, obtained from multilingual training data, to identify subtle linguistic patterns considering the specific context of segments. Consequently, an over-specific behavior is not desired. Given the above considerations, we decided to improve the prompt by using a two-shot only approach. One example was selected to instruct GPT-4o to recognize comma splice Fluency errors, which were among the common Fluency errors in the development dataset. The other example aimed to guide GPT-4o in identifying Omission errors, in which the marked text and error span were not from the target English translation, but from the source Chinese sentence.

Additionally, we removed the part in which we required LLM to generate an explanation for each error. Figure 4.7 is the improved version of the prompt[4]

4.4.3 Result analysis

We analyzed the results in two Setups. In Setup 1, we kept errors of all severity scales. In Setup 2 we removed errors of severity scales 1 and 2. From Figure 4.8 we can see that by removing low severity errors, Setup 2 had a more similar error number distribution with human annotations.

After testing different mappings of the severity scale numbers to Major or Minor labels, we found that the highest quality score correlations occurred when 3 and 4 were mapped to Minor and 5 to Major. We used this mapping for both Setups. Table 4.6 shows the results of two setups.

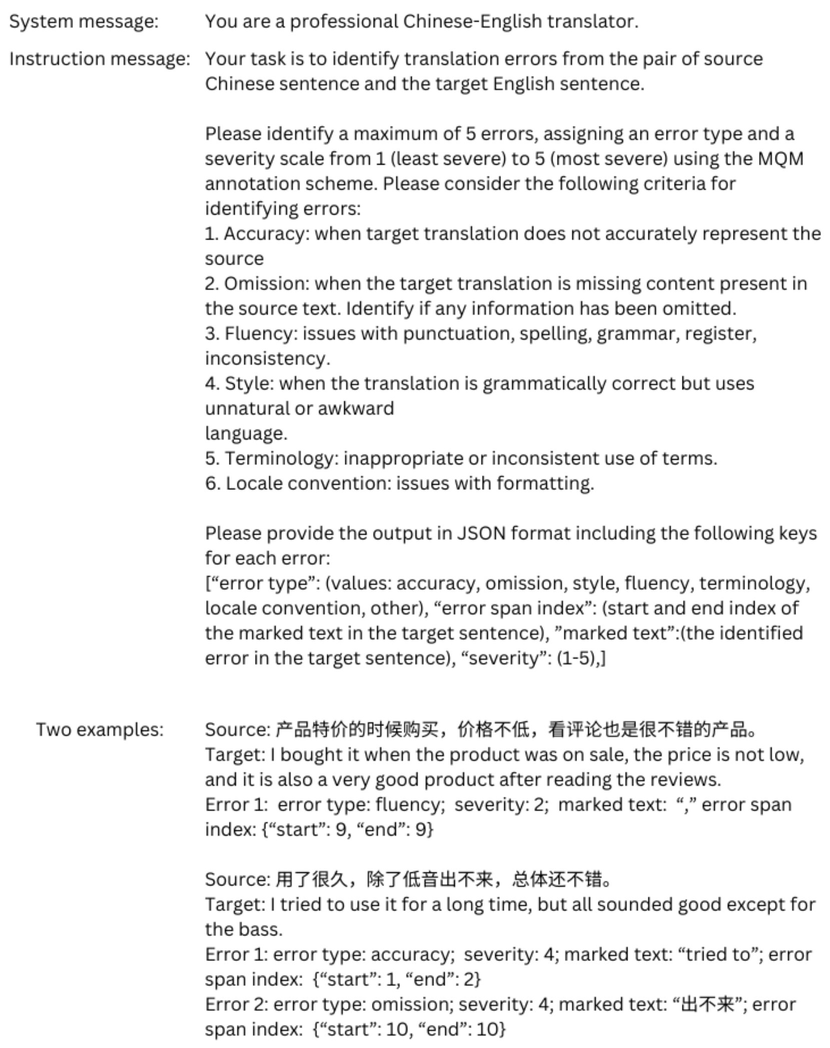
Number of parsable output

More parsable segment annotations were obtained in Setup 2, which was reasonable because fewer errors led to fewer problematic annotations that were due to key parsing failures.

Severity

In Setup 2, for 942 segments in the development dataset, human annotations had 1490 errors (with 318 Major and 1172 Minor), while in the 1307 errors that remained,

[4]The format of the actual prompt in used was slightly different to better fit the format provided in this OpenAI cookbook: `https://github.com/openai/openai-cookbook/blob/main/examples/How_to_format_inputs_to_ChatGPT_models.ipynb`; access date: April 25th, 2024.



System message: You are a professional Chinese-English translator.

Instruction message: Your task is to identify translation errors from the pair of source Chinese sentence and the target English sentence.

Please identify a maximum of 5 errors, assigning an error type and a severity scale from 1 (least severe) to 5 (most severe) using the MQM annotation scheme. Please consider the following criteria for identifying errors:
1. Accuracy: when target translation does not accurately represent the source
2. Omission: when the target translation is missing content present in the source text. Identify if any information has been omitted.
3. Fluency: issues with punctuation, spelling, grammar, register, inconsistency.
4. Style: when the translation is grammatically correct but uses unnatural or awkward
language.
5. Terminology: inappropriate or inconsistent use of terms.
6. Locale convention: issues with formatting.

Please provide the output in JSON format including the following keys for each error:
[“error type”: (values: accuracy, omission, style, fluency, terminology, locale convention, other), “error span index”: (start and end index of the marked text in the target sentence), ”marked text”:(the identified error in the target sentence), “severity”: (1-5),]

Two examples: Source: 产品特价的时候购买，价格不低，看评论也是很不错的产品。
Target: I bought it when the product was on sale, the price is not low, and it is also a very good product after reading the reviews.
Error 1: error type: fluency; severity: 2; marked text: “,” error span index: {“start”: 9, “end”: 9}

Source: 用了很久，除了低音出不来，总体还不错。
Target: I tried to use it for a long time, but all sounded good except for the bass.
Error 1: error type: accuracy; severity: 4; marked text: “tried to”; error span index: {“start”: 1, “end”: 2}
Error 2: error type: omission; severity: 4; marked text: “出不来”; error span index: {“start”: 10, “end”: 10}

Figure 4.7: Two-shot prompt for sub-experiment 4 improved upon previous prompting experiments

there were 367 Major and 940 Minor errors. Both the absolute numbers of Major and Minor errors and the ratio of Major to Minor labels were closer to human annotations compared to Setup 1.

By implementing a scale mapping method for severity labels, we managed to increase the severity f1 score: in sub-experiment 3, GPT-4o obtained a score of 0.51 (see Table 4.2), while in Setup 1 and 2, the scores were 0.65 and 0.62.

Error span

Setup 2 had a higher error span f1 than Setup 1. In terms of recall and precision scores, Setup 1 had a higher recall score than Setup 2, which was reasonable because more errors were reserved for evaluation of error span matching. Meanwhile, the precision of Setup 2 was higher than Setup 1, which revealed a better match between higher severity scale errors (3, 4 and 5) and human annotations.

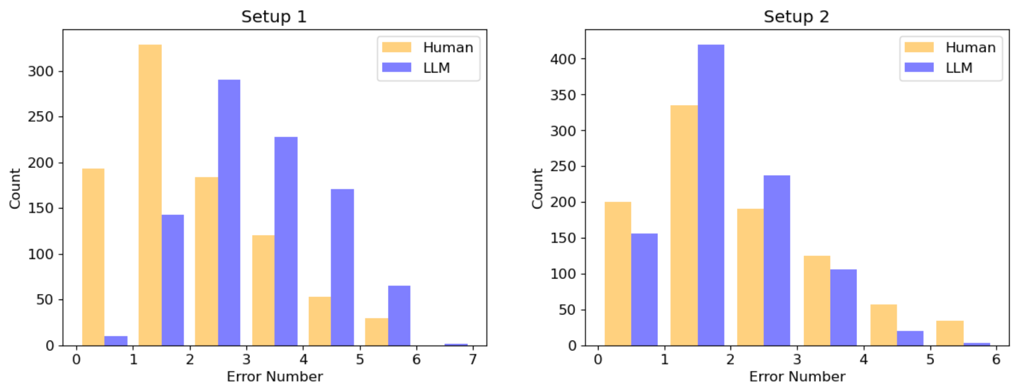


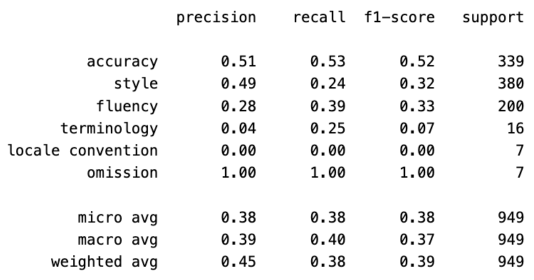
Figure 4.8: Sub-experiment 4: error number distribution of two setups

Setup	Parsable output	Num. of outputted errors (Major/Minor)	Error span f1	Severity - macro avg f1	Error. type - macro avg f1	Quality score - Pearson	Quality score - Spearman
Setup 1	909	2428 (354/2074)	0.33 (r 0.458; p 0.334)	0.65	0.37	0.492	0.376
Setup 2	942	1307 (367/940)	0.343 (r 0.379; p 0.407)	0.62	0.35	0.504	0.404

Table 4.6: Sub-experiment 4: results of two setups

Error type

In our two-shot setting, we added two examples: a common comma splice Fluency error and an Omission error. When evaluating error type, Omission was added as one of the top-category labels.



```
                   precision    recall  f1-score   support

          accuracy      0.51      0.53      0.52       339
             style      0.49      0.24      0.32       380
           fluency      0.28      0.39      0.33       200
       terminology      0.04      0.25      0.07        16
 locale convention      0.00      0.00      0.00         7
          omission      1.00      1.00      1.00         7

         micro avg      0.38      0.38      0.38       949
         macro avg      0.39      0.40      0.37       949
      weighted avg      0.45      0.38      0.39       949
```

Figure 4.9: Error type classification report: Setup 1

In Setup 1 (see Figure 4.9 classification report), there were 7 support cases for the Omission label with an f1 score of 1, suggesting that GPT-4o was able to identify Omission errors, extracting text from the source Chinese sentence. For Fluency, the f1 score increased from 0.29 to 0.33 (with recall/precision from 0.37/0.23 to 0.39/0.28). The results showed a potential positive influence when two examples were provided.

For all error type labels, Setup 1 in this sub-experiment achieved a higher macro avg f1 of 0.37 compared to 0.26 in the previous sub-experiment 3 of GPT-4o. The weighted average was 0.39, which was also higher than the 0.35 in sub-experiment 3. This might indicate adding detailed explanations had improved GPT-4o's performance in assigning correct labels.

Quality score correlation

Both Setup 1 and 2 obtained higher Pearson and Spearman correlations than the result from sub-experiment 3 (see Table 4.2). In all prompting experiments, Setup 2 exhibited the highest correlations among all LLMs. In the MQM annotation to quality score formula defined in Section 2.1.3, Major errors were penalized 5 times more than Minor errors. In this sub-experiment, we implemented a severity scale to Major/Minor mapping, which may have increased the correlations.

4.4.4 Sub-conclusion

In this sub-experiment, we improved our prompt by adding: 1) the Omission error type; 2) detailed explanations for each error type; 3) one Fluency example and one Omission example. Additionally, instead of Major/Minor labels, we prompted GPT-4o for a scale number of 1-5 for severity. Then we devised a scale number to Major/Minor labels mapping method. As a result, we found that: GPT-4o was able to identify Omission Errors; the performance for error type classification improved slightly; the quality score correlations increased.

We decided to use Setup 2 to annotate segments for the QE model training and named it PPbMQM (prompt-pattern-based-MQM), for it having a similar error number distribution with human annotations and obtaining the highest correlations. This prompting technique includes:

- The implementation of the GPT-4o model
- The prompt used in this sub-experiment (showed in Figure 4.7)
- The method of handling severity scale number to Major/Minor label (Remove errors with scale 1 and 2; label 3 and 4 as Minor and 5 as Major)

Chapter 5

Training QE model with LLM generated annotations

In this chapter, we introduce the QE model training experiment. First, we apply the PPbMQM prompting technique to annotate the development dataset, the pre-processing of which was described in Section 3.2.1. Then we use the quality scores deriving from human annotations to train the baseline QE model, and scores deriving from PPbMQM annotations to train the PPbMQM QE model. The goal of this experiment is to explore how LLM annotations will influence the downstream QE model in comparison with the baseline model.

5.1 Training data annotation

We annotated 20703 segments by GPT 4o applying the PPbMQM prompting technique, resulting in 19869 parsable MQM annotations. For these segments we further conducted a fingerprint analysis, the result of which is shown in Table 5.1. Compared to other scores, the quality score Pearson correlations of two fingerprints showed a larger difference, with one of 0.399 and the other of 0.478. It was worth noticing that there was an imbalance in segment numbers between two fingerprints: one with 19155 segments and the other with 714. Considering the imbalance of segments and not-so-big correlation difference, it was difficult to infer that one fingerprint performed better than the other. The overall performance of GPT-4o can still be considered stable.

GPT-4o system-fingerprint	Parsable output	Error span F1	Severity - Macro avg F1	Error type - Macro avg F1	Quality score - Pearson	Quality score - Spearman
fp_319be4768e	19155	0.276	0.47	0.29	0.399	0.274
fp_aa87380ac5	714	0.289	0.49	0.28	0.478	0.239

Table 5.1: The result of two fingerprints

5.2 Model training

5.2.1 Baseline human annotation model

The baseline reference-less QE model of this project was built on the COMET neural framework[1]. We kept the default setting, using XLM-RoBERTa as the encoder and xlm-roberta-large as the pre-trained model. The model was trained on 19869 segments with human annotations. The detailed setup of this model can be found in Appendix B.

5.2.2 PPbMQM QE model

The PPbMQM QE model was trained on the same 19869 segments with quality scores deriving from GPT 4o annotations which was described in Section 5.1. In this model, except for the learned quality scores, the rest setup remained the same as the baseline model.

5.3 Model testing

The test set, which was introduced in Section 3.2.1, contained 5000 segments. As described in Section 3.2, the domains of test set data include news, e-commerce, and manuals. The training data, on the other hand, covers all domains of the test set except for the manuals domain. Table 5.2 shows the result on the test set.

Model	Pearson	Spearman	Kendall
Baseline	0.470	0.410	0.290
PPbMQM QE	0.513	0.381	0.272

Table 5.2: The results of the baseline model and PPbMQM QE model

5.3.1 Result analysis

On the test set, the PPbMQM QE model achieved a higher Pearson correlation score than the baseline model (0.513 compared to 0.470), but the Spearman and Kendall correlation scores were slightly lower than the baseline model. Nevertheless, this further revealed that the segments annotated by the PPbMQM prompt technique could achieve human-annotation level performance and be used in downstream QE model training.

Number of segments	Gold score range	Pearson	Spearman	Kendall
2592	0.9 - 1.0	0.019	0.020	0.016
910	0.8 - 0.9	0.063	0.077	0.053
1304	less than 0.8	0.434	0.425	0.294

Table 5.3: Baseline model: bucket analysis

Furthermore, we conducted bucket analysis for the baseline model and PPbMQM QE model: we divided the test set into buckets based on the ranges of quality scores.

[1]Github: `https://github.com/Unbabel/COMET`

Number of segments	Gold score range	Pearson	Spearman	Kendall
2592	0.9 - 1.0	-0.157	-0.128	-0.085
910	0.8 - 0.9	-0.005	0.016	0.011
1304	less than 0.8	0.523	0.526	0.370

Table 5.4: PPbMQM QE model: bucket analysis

The test set was bucketed into three: segments with quality scores ranging from 0.9 - 1 (high quality), 0.8 - 0.9 (medium quality), and lower than 0.8 (low quality). The result can be found in Tables 5.3 and 5.4. Two models performed better on low-quality segments than high quality. However, the PPbMQM QE model showed larger differences in three buckets than the baseline model, revealing a more nonmonotonic relationship between variables. We assumed the reason was that GPT-4o tended to be stricter than humans when annotating which amplified the effect of errors, resulting in quality scores of a larger range that increased the non-monotonic relationship between the two variables.

5.3.2 Sub-conclusion

From this experiment, we conclude that segments annotated by PPbMQM can be used to train a downstream QE model that can even surpass models trained on human annotations. In addition, our PPBMQM QE model correlated better with segments of lower quality.

Chapter 6

Discussion

The QE model experiment shows that the PPbMQM QE model achieved better Pearson correlation with human judgment than the baseline model trained on human annotations. This provides strong evidence of the feasibility of using LLMs annotations for downstream QE model training.

6.1 Humans are not always the best annotators

In Section 3.2.2, we conducted a correlation analysis of the agreement on the same segments between human annotators. The annotations claimed to be performed by experts (Freitag et al., 2021), but we found that the Pearson correlations were very low, and in some cases, even negative. As previously mentioned, MQM annotations require experts proficient in both languages. The low level of agreement highlights several issues related to the Chinese-to-English annotation task:

First, different annotators may have different criteria when identifying specific translation errors, involving their subjectivity. Furthermore, obtaining objective annotations becomes more challenging when the annotation requires domain-specific knowledge or when the source sentence itself is ambiguous.

Secondly, the low agreement between annotators may lead us to question annotators' proficiency in both languages. In Section 4.3.2, we examined the extra errors that were identified only by LLMs. The non-hallucination cases show that LLMs are adept at identifying errors related to natural word usage. The language pair we study in this project is Chinese to English. These two languages are from different language families and have few similarities in linguistic features, such as syntax, morphology, and vocabulary. These differences make it more challenging to recruit annotators. Additionally, Chinese is considered a high-resource language with a vast amount of internet data for LLM training. Given these factors, we encourage researchers not to be discouraged by the seemingly low correlation between LLM annotations and human annotations. Instead **researchers should always evaluate the quality of human annotations, as LLMs may produce better results**.

Third, in regard to hallucinations, our analysis in Section 4.3.2 revealed that in our samples, hallucinations were not particularly prevalent, especially for GPT-4 models. Additionally, when utilizing annotations generated by language models to train a score prediction QE model, the tendency of hallucinating Minor errors would not make such a big negative impact. This is because Minor errors are penalized less severely than Major errors when converting annotations to quality scores.

Despite the tendency for hallucinations, LLMs may still be better annotators than humans for the Chinese-to-English language pair.

6.2 Reproducibility

6.2.1 LLMs stability

In Section 4.3.5, we tested on 400 segments by using a basic prompt variant. In Section 5.1, we compared the results from two different fingerprints of GPT-4o. Both results indicate that GPT-4o can perform in a consistent way for annotation tasks.

6.2.2 Experiment setup and open resources

We already tested that GPT-4o can perform in a stable manner for our annotation task. In our project, to ensure reproducibility, we have made several attempts to present as many experiment details as possible: for every prompt experiment, all setups, including seed values, can be found in Appendixes A and B. For every dataset, website, and GitHub repository, all links were attached as footnotes.

6.2.3 Proprietary or open-source LLM?

In sub-experiment 3, the open-source model LLaMA 3 obtained the highest quality score Pearson and Spearman correlations, which demonstrates its capabilities for QE tasks. Nevertheless, it also had an unstable API and tended to hallucinate more often than GPT-4 models. Nonetheless, as the capacity of open LLMs continues to grow, we are positive about employing them in future MQM annotation research.

Our prompting technique PPbMQM depends on GPT-4o, a 'black-box' proprietary model (Kocmi and Federmann, 2023a). Since the training and deployment details for proprietary models are usually not transparent to the public, there have been concerns about using such models in the academic environment. Yet undeniably, GPT models demonstrate promising capabilities for our annotation tasks and stabilities of APIs with traceable backend fingerprints.

We argue that alongside the debate over proprietary or open-source LLMs, consistent efforts should be made to establish a strong methodological framework for prompt engineering. A systematic and robust methodology approach will enable more advanced capabilities of rapidly evolving LLMs.

6.3 Limitation and ethical issues

- The baseline model and PPbMQM QE model were both trained with only one initialization. Future efforts should be made to train with multiple initializations to obtain a sufficient sample for the statistical test, strengthening the experimental evidence (Ulmer et al., 2022).
- We only investigated the Chinese-to-English language pair, which are both high-resource languages. We cannot infer our method performs well for other low-resource language pairs.
- Ethical statement: in our project, the LLMs we used may have biases that perpetuate stereotypes.

6.4 Future research

6.4.1 Flexible prompt development for other language pairs

Future research can focus on adapting our prompting technique to other language pairs. We assume that our prompting technique can have decent performance for English and another language that is high or at least medium resource. In addition, before testing on a large number of segments, we encourage researchers to interact with LLMs first using various prompting patterns such as *the question refinement pattern* to explore their behaviors on other language pairs.

6.4.2 From score prediction to fine-grained error detection QE

In future research, we encourage researchers to use LLM-generated annotations to train downstream QE models with smaller sizes, instead of directly prompting models with sizes of GPT-4 for QE. To make full use of the rich information provided by MQM annotations, one possible approach is to train a downstream QE model that can predict the error spans and their severity labels.

6.4.3 Combine output from different LLMs

In our project, we prompted three LLMs for MQM annotations and compared the results of each with human annotations. Future research can further compare one LLM's result with other LLMs, e.g., obtain the overlapped errors from LLMs to examine if these LLMs have similar tendencies towards specific types of hallucinations.

Chapter 7

Conclusion

In this project, we investigated LLMs' behaviors on the MQM annotation task and developed a prompting technique PPbMQM. We conclude that for the Chinese-to-English language pair, it is feasible to use LLMs as annotators. From the result of the QE model experiment, we see that the model utilizing LLM annotations can even outperform the model trained on human judgment data.

We applied prompt patterns in prompt design and developed the prompt iteratively based on a series of prompting experiments. This approach strengthens the methodological aspect of prompt development and increases prompts' structural component and reusability.

The prompting technique PPbMQM can be used directly for MQM annotation tasks, or further refined and adapted to other LLMs and other language pairs. Future research should focus on training downstream QE models that can predict more labels such as error spans and severity, making full use of the rich information provided by MQM annotations. Meanwhile, consistent efforts should be made consistently to 1) enhance the methodology for prompt engineering to adapt to the rapidly evolving capabilities of LLMs and 2) target low-resource language pairs.

Appendix A

Prompting experiment setup

A.1 Sub-experiment 1

LLM	Parameters
LLaMA 3	min_token= 0, max_token = 512, temperature = 0.6, top_p: 0.9, presence_penalty = 1.15, seed = 240425
GPT-3.5 Turbo; GPT-4.0 Turbo; GPT-4o	seed = 240425, max_tokens = 512, temperature = 0.2, top_p = 1,frequency_penalty =0, presence_penalty= 0

Table A.1: Experiment setup: Sub-experiment 1

A.2 Sub-experiment 2 & 3

LLM	Parameters
LLaMA 3	min_token= 0, max_token = 1024, temperature = 0.6, top_p: 0.9, top_k = 0, length_penalty = 1, presence_penalty = 1.15, seed = 240425
GPT-4.0 Turbo; GPT-4o	seed = 240425, max_tokens = 1024, temperature = 1, top_p = 1,frequency_penalty =0, presence_penalty= 0

Table A.2: Experiment setup: Sub-experiment 2 & 3

A.3 Sub-experiment 4

LLM	Parameters
GPT-4o	seed = 240425, max_tokens = 1024, temperature = 1, top_p = 1,frequency_penalty =0, presence_penalty= 0

Table A.3: Experiment setup: Sub-experiment 4

Appendix B

Baseline model experiment setup

Hardware:
Machine: x86_64
Platform: Linux-6.1.85+-x86_64-with-glibc2.35

Below are the hyper-parameter from the **hparam.yaml** file that was automatically generated containing configuration details:

```
activations: Tanh
batch_size: 4
class_identifier: referenceless_regression_metric
dropout: 0.1
encoder_learning_rate: 1.0e-06
encoder_model: XLM-RoBERTa
final_activation: null
hidden_sizes:
- 2048
- 1024
keep_embeddings_frozen: true
layer: mix
layer_norm: false
layer_transformation: sparsemax
layerwise_decay: 0.95
learning_rate: 1.5e-05
load_pretrained_weights: true
loss: mse
nr_frozen_epochs: 0.3
optimizer: AdamW
pool: avg
pretrained_model: xlm-roberta-large
```

In addition, the seed value used for both models was 12 (the seed_everything parameter).

Appendix C

Resources

Below are open resources we used in this project:

COMET framework: `https://github.com/Unbabel/COMET`

Expert-based Human Evaluations of WMT General MT: `https://github.com/google/wmt-mqm-human-evaluation`

OpenAI cookbook *How to make your completions outputs consistent with the new seed parameter*: `https://cookbook.openai.com/examples/reproducible_outputs_with_the_seed_parameter`

OpenAI cookbook *How to format inputs to ChatGPT models*: `https://github.com/openai/openai-cookbook/blob/main/examples/How_to_format_inputs_to_ChatGPT_models.ipynb`

MQM official website: `https://themqm.org/error-types-2/typology/`

Bibliography

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

F. Blain, C. Zerva, R. Rei, N. M. Guerreiro, D. Kanojia, J. G. de Souza, B. Silva, T. Vaz, Y. Jingxuan, F. Azadi, et al. Findings of the WMT 2023 shared task on quality estimation. In *Proceedings of the Eighth Conference on Machine Translation*, pages 629–653, 2023.

A. Burchardt. Multidimensional quality metrics: a flexible system for assessing translation quality. In *Proceedings of Translating and the Computer 35*, 2013.

A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.747. URL `https://aclanthology.org/2020.acl-main.747`.

P. Fernandes, D. Deutsch, M. Finkelstein, P. Riley, A. F. Martins, G. Neubig, A. Garg, J. H. Clark, M. Freitag, and O. Firat. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. *arXiv preprint arXiv:2308.07286*, 2023.

M. Freitag, G. Foster, D. Grangier, V. Ratnakar, Q. Tan, and W. Macherey. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474, 2021.

M. Freitag, R. Rei, N. Mathur, C.-k. Lo, C. Stewart, E. Avramidis, T. Kocmi, G. Foster, A. Lavie, and A. F. Martins. Results of WMT22 metrics shared task: Stop using BLEU–neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 46–68, 2022.

F. Kepler, J. Trénous, M. Treviso, M. Vera, and A. F. T. Martins. OpenKiwi: An open source framework for quality estimation. In M. R. Costa-jussà and E. Alfonseca, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 117–122, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3020. URL `https://aclanthology.org/P19-3020`.

T. Kocmi and C. Federmann. GEMBA-MQM: Detecting translation quality error spans with GPT-4. *arXiv preprint arXiv:2310.13988*, 2023a.

T. Kocmi and C. Federmann. Large language models are state-of-the-art evaluators of translation quality. *arXiv preprint arXiv:2302.14520*, 2023b.

T. Kocmi, C. Federmann, R. Grundkiewicz, M. Junczys-Dowmunt, H. Matsushita, and A. Menezes. To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. *arXiv preprint arXiv:2107.10821*, 2021.

T. Kocmi, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, T. Gowda, Y. Graham, R. Grundkiewicz, B. Haddow, et al. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, 2022.

T. Kocmi, E. Avramidis, R. Bawden, O. Bojar, A. Dvorkovich, C. Federmann, M. Fishel, M. Freitag, T. Gowda, R. Grundkiewicz, et al. Findings of the 2023 conference on machine translation (wmt23): Llms are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42, 2023.

S. Lee, J. Lee, H. Moon, C. Park, J. Seo, S. Eo, S. Koo, and H. Lim. A survey on evaluation metrics for machine translation. *Mathematics*, 11(4):1006, 2023.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

A. Lommel, H. Uszkoreit, and A. Burchardt. Multidimensional quality metrics (MQM): A framework for declaring and describing translation quality metrics. *Tradumàtica*, (12):0455–463, 2014.

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

M. Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395, 2015.

R. Rei, C. Stewart, A. C. Farinha, and A. Lavie. COMET: A neural framework for MT evaluation. *arXiv preprint arXiv:2009.09025*, 2020.

R. Rei, M. Treviso, N. M. Guerreiro, C. Zerva, A. C. Farinha, C. Maroti, J. G. De Souza, T. Glushkova, D. M. Alves, A. Lavie, et al. CometKiwi: IST-Unbabel 2022 submission for the quality estimation shared task. *arXiv preprint arXiv:2209.06243*, 2022.

R. Rei, N. M. Guerreiro, J. Pombal, D. van Stigt, M. Treviso, L. Coheur, J. G. de Souza, and A. F. Martins. Scaling up CometKiwi: Unbabel-ist 2023 submission for the quality estimation shared task. *arXiv preprint arXiv:2309.11925*, 2023.

T. Sellam, D. Das, and A. P. Parikh. BLEURT: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

L. Specia and K. Shah. Machine translation quality estimation: Applications and future perspectives. *Translation quality assessment: from principles to practice*, pages 201–235, 2018.

H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

D. Ulmer, E. Bassignana, M. Müller-Eberstein, D. Varab, M. Zhang, R. Van Der Goot, C. Hardmeier, and B. Plank. Experimental Standards for Deep Learning in Natural Language Processing Research. *arXiv preprint arXiv:2204.06251*, 2022.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.

S. Xenouleas, P. Malakasiotis, M. Apidianaki, and I. Androutsopoulos. SumQE: a bert-based summary quality estimation model. *arXiv preprint arXiv:1909.00578*, 2019.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.