Generative AI for Technical Writing: Comparing Human and LLM Assessments of Generated Content

Karen de Souza
University of Eastern Finland

Alexandre Nikolaev
University of Eastern Finland

Maarit Koponen
University of Eastern Finland

{kpatteri, alexandre.nikolaev, maarit.koponen} @uef.fi

Abstract

Large language models (LLMs) have recently gained significant attention for their capabilities in natural language processing (NLP), particularly generative artificial intelligence (AI). LLMs can also be useful tools for software documentation technical writers. We present an assessment of technical documentation content generated by three different LLMs using retrieval-augmented technology (RAG) with product documentation as a knowledge base. The LLM-generated responses were analyzed in three ways: 1) manual error analysis by a technical writer, 2) automatic assessment using deterministic metrics (BLEU, ROUGE, token overlap), and 3) evaluation of correctness by LLM as a judge. The results of these assessments were compared using a Network Analysis and linear regression models to investigate statistical relationships, model preferences, and the distribution of human and LLM scores. The analyses concluded that human quality evaluation is more related to the LLM correctness judgment than deterministic metrics, even when using different analysis frameworks.

1 Introduction

Technical communication means creating content based on factual data, as consistently and clearly as possible, so that users can easily understand complex technical concepts. Various professionals are involved in it, such as technical translators, developers, information architects, and technical writers (Society for Technical Communication, n.d.). Technical documentation provides specialized and task-oriented information for the user on how to use and interact with a given product. It is not feasible to cover all possible use cases; instead, the focus should be on the main functionalities or use cases to maintain objectivity. (Swarts, 2018).

LLMs can be useful not only for code generation but also for technical writing because they can simplify the documentation process by generating drafts when prompted with code snippets. This can facilitate the work of technical writers and reduce the effort needed for research. A good model could lower the technical barriers, automate lengthy tasks, and act as an extra solution for their problems (Evtikhiev et al., 2023). However, due to several facts, one of them possibly being outdated training data, LLM-based chatbots can also hallucinate information that does not accurately reflect reality. An alternative to tackle this issue is RAG. It allows external data to be incorporated into the model, which improves its ability to provide more relevant or up-to-date responses, based on the data used to implement the RAG method (Gao et al., 2023).

This study leverages the content produced by a chatbot using multiple LLMs (GPT-3.5 Turbo, GPT-4.0, and Mistral AI 7B) and RAG technology on a specific topic: Network as Code (NaC) technical product documentation. Briefly, NaC simplifies the programming of networks, such as automatically adjusting streaming capabilities and improving bandwidth, for example, for online games or concert streaming (Nokia, 2024).

The LLM responses were evaluated by a technical writer using an error-typology framework and an LLM as a judge based on answer correctness. The aim of this paper is to understand how different (automatic and human) evaluations are distributed according to the attributed scores and how, or if, they relate. Additionally, the ultimate purpose is not to show which evaluation is the best but to offer insights on how performing different analyses, for instance, automatic (quantitative)

Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025), pages 661–679
March 3-4, 2025

and qualitative can complement one another.

We first discuss relevant work on evaluating LLM output quality (Section 2), and then present the dataset and evaluation approaches used in this study (Section 3). We report the comparison of human, deterministic, and LLM evaluations using linear regression models, score distributions Section 4.1), and Network Analysis (Section 4.2). Finally, we discuss the implications of the comparison (Section 5) and present the conclusions and directions for future work (Section 6).

2 Related work

2.1 Leveraging Translation Quality Evaluation for LLM Analysis

For evaluating the output of LLMs, frameworks and methods have been applied from other contexts, such as (machine) translation quality evaluation. Deterministic metrics, such as Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002), which are based on the comparison of a "gold-standard" human translation and an automatically generated "hypothesis", have traditionally been used in the machine translation (MT) field. However, recent advancements in the quality of generative systems have led to increasing skepticism about their reliability, often showing little or no correlation with human assessment (Freitag et al., 2022). BLEU, as well as other metrics such as Recall-Oriented Understudy for Gisting Evaluation (ROUGE), chrF, and token overlap, are also used for evaluating LLM generated output (Zhang and Antonante, 2023).

However, deterministic metrics are not standalone solutions and should be combined with human qualitative evaluation. High-quality and granular standards can again be provided by translation quality evaluation frameworks where professionals manually check and annotate errors according to their severity following the specified error criteria (Fernandes et al., 2023). Error typologies, such as the Dynamic Quality Framework (DQF) and the Multidimensional Quality Metrics (MQM), provide detailed quality criteria to evaluate translations based on categories for accuracy, fluency, style, and design, and attribute point deductions or penalties according to the severity of the error identified (Castilho et al., 2018). The combined DQF-MQM framework defines error categories and sub-categories, penalties, and thresholds that can be adjusted by professional evaluators depending on their current work needs (Castilho et al., 2018). This framework is commonly used in the field and is considered a reliable methodology for translation quality evaluation. Due to its focus on specific textual features and adjustable categories, it can also be applied to texts independently of their type, such as technical content creation.

2.2 LLMs as quality evaluators

Recent work has also investigated how LLMs can be used as quality evaluators and be prompted (using instructions) to analyze the output quality of MT and generative AI. Kocmi and Federmann (2023a) used a GPT-based evaluation metric called GEMBA-DA for translation quality assessment. Their encoder-only and encoder-decoder language models used supervised data consisting of human gold-standard evaluations in the form of 0 to 100 direct assessments from the WMT22 Metrics shared task. Their approach used zero-shot prompts requesting different LLMs to score each source-target pair on a scale from 0 to 100. The study concluded that this evaluation method was only successful in GPT-3.5 and larger models (Kocmi and Federmann, 2023a). Later work by Kocmi and Federmann (2023b) created an improved model called GEMBA-MQM, which uses GPT-4 to assess translation quality error spans following the MQM framework criteria as a reference. The researchers used a more detailed few-shot prompting technique providing the LLM with the same instructions a human evaluator would receive. The work concludes that the GEMBA-MQM model assessment has higher correlations with human judgment because of a common translation quality evaluation framework. It also compares the LLM scores against deterministic metrics in the scientifically related literature, such as BLEU and chrF, to reach this conclusion (Kocmi and Federmann, 2023b).

These studies indicate that LLMs can provide state-of-the-art quality evaluations that are more correlated with human judgment when they learn or are fine-tuned with human evaluation data, such as the ones produced with the MQM framework (Fernandes et al., 2023). Other approaches, such as those within Continuous-eval packages evaluate LLM-generated text and code with granular or holistic approaches, including quality and deterministic assessment of generative AI content (Zhang and Antonante, 2023). How to evaluate the quality of generative AI content remains, however, an open question. The present study further explores this question by evaluating texts generated for technical documentation purposes and comparing the evaluations provided by deterministic metrics and an LLM to a manual assessment by a professional technical writer.

3 Material and Methods

3.1 Dataset

The outputs analyzed in this study were generated with Nokia's chatbot tool which is internally used in the company. Nokia authorized its use for this research. This application allows the ingestion of documents, such as Markdown files as input so the chatbot can provide better responses based on the knowledge base provided. RAG can be implemented through technologies such as LangChain, which enables semantic search on relevant content (LangChain, 2023). Further details of this implementation cannot be revealed as the tool is proprietary. The prompts and responses relate strictly to the public documentation of NaC. The dataset includes 12 prompts, each generating 6 different types of responses from three distinct LLMs: GPT-3.5-turbo-16k, GPT-4-1106-preview (OpenAI, 2023) and Mistral AI 7B (Mistral AI, 2023) with two different temperature sets to either 0.4, a more deterministic tone, or 0.7, a more creative one. The chatbot was set to a maximum of 2,048 tokens to avoid overly extensive generated content. The data were collected and analyzed as part of a Master's thesis project in the spring of 2024 (de Souza, 2024).

Zero and few-shot prompt-engineering are the chosen techniques, in which zero shots are simple prompts with no further instructions on how the response should be, and the few-shot ones provide a simple and limited number of examples or instructions for the response (DAIR.AI, n.d.). For instance, in few-shot prompts that requested an LLM to generate a documentation page for a given code snippet about a NaC functionality, instructions were given on how to organize the documentation page in Markdown language, and the main technical concepts to be clarified by the LLM were specified in the prompt.

3.2 Manual evaluation

The responses generated by the different LLMs were analyzed by a professional technical writer according to the DQF-MQM framework (TAUS, n.d.) using error categories adapted for the purpose of prompt analysis. A discussion of the error analysis is outside of the scope of this paper, but the list of categories and error evaluations can be found in a GitHub repository.[1] A more detailed description is given in de Souza (2024). Based on the errors identified, point deduction penalties were applied according to the severity of the error as follows:

- 0 - no points deducted, in which case the response is correct.
- 0.25 - deducted when errors do not lead to loss of meaning or major confusion.
- 0.5 - deducted when errors are significantly misleading or confusing.
- 0.75 - deducted if errors could affect the company image, e.g. responses that do not include or disregard privacy reminders.

More than one penalty could be applied to the same response if multiple errors were identified in the same response. After the penalties, each response received a total score ranging from 0 (totally irrelevant responses) to 1 (totally correct and relevant responses).

3.3 LLM quality analysis

A qualitative analysis was also performed with the Continuous-eval LLM-based correctness package, which can implement different LLM models to evaluate answer correctness (Relari, 2023). The chosen judge model for this study is GPT-4-1106-preview. The code package allows importing an LLM, which runs through a JSONL file with multiple lines, each containing a set of prompt, response, and related ground truth contexts. The generated responses are evaluated according to their relevance to the prompts, and a total score is given to each response ranging from 0 to 1 as follows:

- 0 indicates the response is totally irrelevant to the prompt.
- 0.25 for responses that are relevant to the prompt but contain major errors.

[1]https://github.com/kjp-souza/tech-writing-LLM-human-evaluation

- 0.5 for responses that are relevant to the prompt but are partially correct.
- 0.75 is attributed to responses that are relevant to the prompt and correct.
- 1.0 is for responses that are relevant to the prompt and complete.

3.4 Deterministic metrics

The human and LLM quality analyses were compared against deterministic metrics, which evaluate the tokens in both prompt and response sets to leverage their token similarity. This deterministic analysis was done using the deterministic metrics also within the Continuous-eval package and are described as follows (Relari, 2023):

- Token overlap refers to words shared by both sets of texts (ground truth reference and generated response).
- ROUGE-L calculates the longest shared subsequence between the generated response and the ground truth text used as reference.
- BLEU measures how well a generated text matches a reference text using n-gram precision, where each n-gram has a specific weight and applies a brevity penalty to overly short translations.

3.5 Network Analysis

The different analysis results, including the manual human quality evaluation scores and automatically generated deterministic and LLM-based correctness scores, were compared in the R statistical software (R Core Team, 2024). Four different types of analyses were created: Network Analysis, linear regression models, score distribution, and average plots.

The Network Analysis used the `bootnet` package (Epskamp et al., 2018), which allows for estimating the network structure based on the observed data. To obtain a conservative network model, it was necessary to apply the least absolute shrinkage and selection operator (LASSO) method. This helped identify only the most important edges (relationships) in the network, formed by nodes, which contain a descriptive label for the deterministic metrics, human quality evaluation, and LLM correctness evaluation scores. By doing so, over-fitting is avoided so the model can remain interpretable. In other words, Network Analysis edges can capture how changes in one variable relate to changes in another without putting a single one in evidence. Furthermore, a tuning parameter value of 0.5 was chosen for the Extended Bayesian Information Criterion (EBIC), which helps balance model complexity and goodness of fit. A smaller EBIC value indicates a better-fitting model, in this case, the tuning offers a moderate level of regularization, which penalizes very weak connections (edges) in a sparse network (Nikolaev and Bermel, 2022). Once the network was estimated, a threshold was applied (setting the option to true) to remove weak associations based on correlation strength, leaving only meaningful connections in a network that becomes easier to interpret (Nikolaev and Bermel, 2022).

However, to observe if there is any LLM preference in common between human and LLM analysis, linear regression models were created in R programming language and these use dependent variables: Human quality and LLM correctness judgments against multiple independent variables which indicate different LLMs (GPT-3.5, GPT-4 and Mistral). The `sjPlot` library (Lüdecke, 2024) was used to plot and better visualize the relationship between human and LLM evaluation scores and model types (Nikolaev and Bermel, 2022). Additionally, a figure visually representing the distribution of human and LLM scores was created using the `ggplot2` R package (Wickham, 2016a), which is part of the **tidyverse** ecosystem (Wickham et al., 2019) and used to design graphics according to the grammar of graphics approach (Wickham, 2016b).

4 Results

4.1 Comparison of human and LLM scores

Figure 1 shows the distribution of scores, detailed in sections 3.2 and 3.3, according to human quality evaluation and LLM correctness judgment. The Y axes contain the response counts, visually representing the distribution, while the X axes represent the score distributions according to both human and LLM judgment columns. The human-evaluation scores plot has several peaks and is more evenly distributed, while the LLM one has 2 major peaks with very few outliers, in which responses got 0 or 1 scores and no 0.5 scores.

This shows that the human evaluation scores varied more on a scale from 0 to 1 while the LLM almost exclusively assigned the scores 0.25 or 0.75 and did not use the score 0.5 at all. Additionally, figure 8 in the appendix A shows the average scores for automatic metrics and human quality evaluation (Human QE), which were also generated using the same R programming language packages.

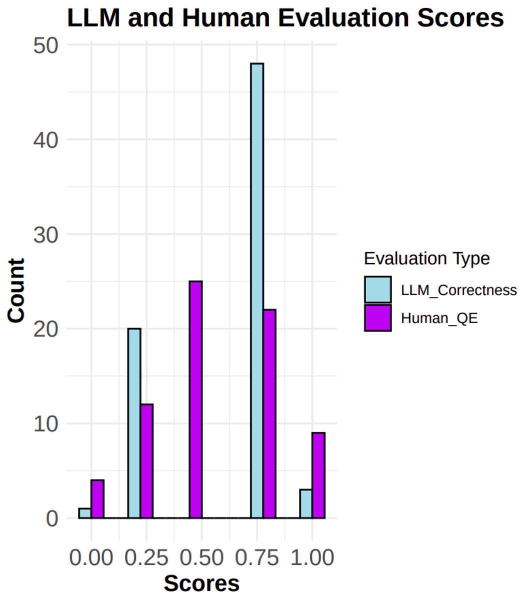


Figure 1: LLM and human scores distribution

Linear regression models were used to analyze the relationships between the dependent variables, human and LLM correctness judgments, and the independent variables, which included multiple language models such as GPT-3.5, GPT-4, and Mistral. We found this approach to be particularly well-suited for handling categorical independent variables, offering significant advantages in terms of flexibility and interpretability. Specifically, it allows the quantification of effect sizes of each language model relative to a designated baseline, either human or LLM judgment, offering deeper insights beyond simple comparisons of group means. Furthermore, the framework accommodates extensions such as continuous predictors and interaction terms, ensuring versatility in our analysis. These features align seamlessly with our research objectives, enabling a nuanced and comprehensive interpretation of the relationships between predictors and outcomes.

Figures 2 and 3 illustrate how well an LLM can predict or explain the quality evaluations by human and LLM (GPT-4-1106-preview) evaluators. The dependent variables are human quality evaluation scores and LLM correctness judgments, while the independent variables represent different LLMs (GPT-3.5, GPT-4, and Mistral), showcasing their influence on human and LLM assessments and enabling a more precise comparison of alignment and similarities in judgment.

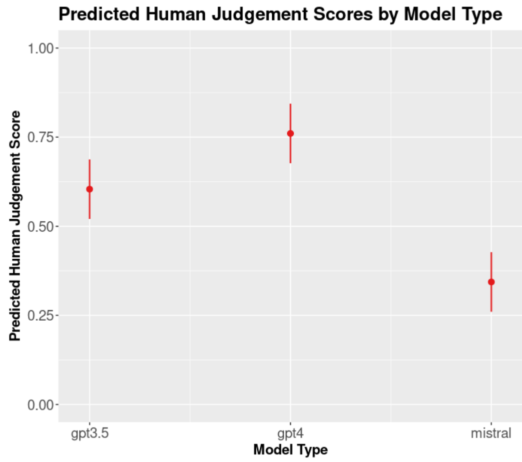


Figure 2: Human analysis linear regression model

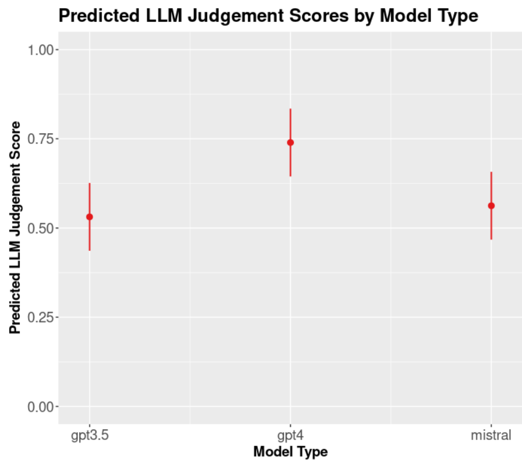


Figure 3: LLM analysis linear regression model

Both figures show that human and LLM evaluations attributed higher or better scores, which surpass 0.75 on a scale from 0 to 1, to content generated by the GPT-4 model. However, GPT-3.5 was better evaluated in the human analysis than in the LLM one. The scores did not drop below the average of 0.50 for GPT-3.5 in human analysis, while they did in the LLM evaluation scores. Mistral models received scores below average in both analyses. However, in LLM analysis, Mistral's scores exceeded 0.60 points while in the human one, it did not even reach 0.50 on average. In conclusion, GPT-4 model responses were better evaluated by both human and LLM analyses, the

GPT-3.5 model was better evaluated only in human analysis and Mistral AI responses were better evaluated only by the LLM one.

4.2 Network Analysis results

As there is no dependent variable in a Network Analysis, it is possible to observe different positive or negative relationships, which indicate behaviors among multiple variables, represented by the evaluation scores, without emphasizing a single one. Figure 4 shows a Network Analysis of all observed variables: qualitative, including human and LLM, and multiple deterministic metrics. It consists of nodes (independent variables) that are connected by edges (statistical relationships). These variables and edges are connected in a spiral format representing statistical relationships between them. The goal is to understand how these variables interact with each other. The code and logic used here follow the same ones used in Nikolaev and Bermel (2022).

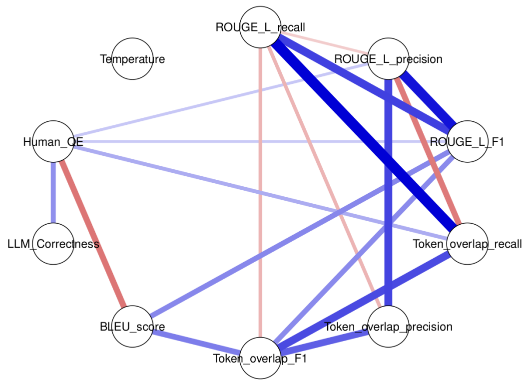


Figure 4: Correlation Network Analysis

The edges in the network (shown as lines) explain the co-variation structure among the observed variables. The blue edges indicate a positive correlation, while the red edges signify a negative one. The color intensity reflects relationship strength. Additionally, this method allows observing the relations between human versus LLM evaluation on one side, and the deterministic correctness metrics relation on the other.

Figure 4 shows that human and LLM evaluations are positively correlated, though the strength of the association is weak to moderate, which can be observed not only in the network but also in the calculated strength of association (0.27). A weight near 0.30 suggests a weak to moderate relationship, while values closer to 1 indicate a stronger association. On the deterministic side, human evaluation is in a weak positive association with the metrics token-overlap recall (0.20), ROUGE-L precision (0.14), and F1 (0.13). This can be connected to content accuracy since the more token overlap there is between the ground truth and the LLM-generated response, the more it is possible to trust it was based on precise and verifiable data. On the other hand, human evaluation has a moderate negative association with BLEU (-0.33). This may be due to the brevity penalty assigned by BLEU to short responses. The LLM temperature node does not influence other evaluations.

Centrality indices were also employed to visualize the relationships between automatic and human evaluation metrics. Figure 5 was generated using the `qgraph` package with the `centralityPlot` function (Epskamp et al., 2018). Nodes represent variables, and edges show statistical relationships in a network. Centrality indices (strength, betweenness, closeness) quantify node importance as standardized z-scores, with higher scores indicating greater centrality or influence relative to the network average. Betweenness represents the number of times a node lies on the shortest path between other nodes, indicating its control over communication in the network; Closeness is the inverse of the sum of distances from a node to all other nodes, measuring how close a node is to all other nodes; and Strength is the sum of the absolute weights of all edges connected to a node, representing how strongly connected a node is to the network.

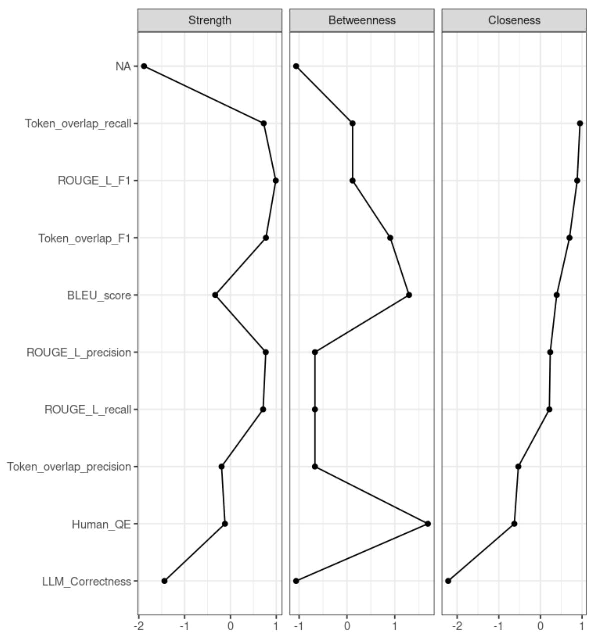
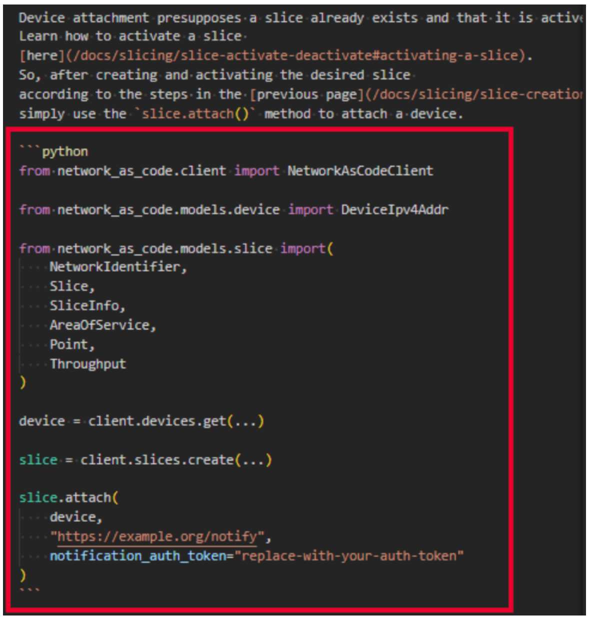


Figure 5: Closeness centrality index

This centrality analysis ordered by closeness reveals distinct patterns. The automatic metrics display more strength and influence due to their quantitative nature while interacting with each other, with token overlap and ROUGE being the strong ones while BLEU is more peripheral. Human QE demonstrates a neutral strength compared to the deterministic metrics. Additionally, the betweenness index shows that it serves as a bridge between LLM evaluation and deterministic metrics. This can be observed through its positive z-score. LLM evaluation remains relatively isolated, suggesting a unique perspective that aligns more closely with human judgment, possibly due to their qualitative nature, than with deterministic metrics. This highlights the complementary nature of qualitative and quantitative evaluations, while human or LLM judgments can provide more holistic assessments, the quantitative metrics offer a granular analysis of quantitative aspects of text quality, such as token overlap. This closeness plot and other centrality ones also ordered by betweenness and strength are included in the appendix A.

4.3 Qualitative analysis example

Figures 6 and 7 show an example of the ground truth reference and a response generated by GPT-4 for the prompt "Can you provide an example of attaching a device to a slice using an SDK?". The human and LLM quality evaluations for this response are given below the figures.



````
Device attachment presupposes a slice already exists and that it is activ
Learn how to activate a slice
[here](/docs/slicing/slice-activate-deactivate#activating-a-slice).
So, after creating and activating the desired slice
according to the steps in the [previous page](/docs/slicing/slice-creatio
simply use the `slice.attach()` method to attach a device.

```python
from network_as_code.client import NetworkAsCodeClient

from network_as_code.models.device import DeviceIpv4Addr

from network_as_code.models.slice import(
    NetworkIdentifier,
    Slice,
    SliceInfo,
    AreaOfService,
    Point,
    Throughput
)

device = client.devices.get(...)

slice = client.slices.create(...)

slice.attach(
    device,
    "https://example.org/notify",
    notification_auth_token="replace-with-your-auth-token"
)
```
````

Figure 6: Ground truth context reference

````
Certainly! Below is an example of how to attach a device to a network
slice using the Network as Code SDK. This example assumes that you have
already created and activated a slice, and that you have a
Network-as-Code client and a device object ready.

```python
from network_as_code.client import NetworkAsCodeClient
from network_as_code.models.device import DeviceIpv4Addr
from network_as_code.models.slice import(
    NetworkIdentifier,
    Slice,
    SliceInfo,
    AreaOfService,
    Point,
    Throughput
)

# Initialize the Network-as-Code client with your token
client = NetworkAsCodeClient(token="<your-application-key-here>")

# Get the device object using its identifier
device = client.devices.get("<device-identifier>")

# Get the slice object using its name or identifier
slice = client.slices.get("<slice-name>")

# Attach the device to the slice
slice.attach(
    device,
    "https://example.org/notify",
    notification_auth_token="replace-with-your-auth-token"
)
```
````

Figure 7: GPT-4 response

- Human evaluation of GPT-4's response:

  "GPT-4 does not use the same style of shortening the code by using ellipsis between parentheses. Instead, it provides example values that work similarly. This LLM did not miss nor omit any imports necessary for the code to work and focused on the main objective of the question. The additional note on the prerequisite of having an active slice is only relevant to the whole documentation page. Since this is an answer to a direct question, points were not deducted for that". Score: 1.0.

- LLM evaluation of GPT-4's response:

  "The answer is relevant to the question and correct. It provides a code example of how to attach a device to a network slice using an SDK, which aligns with the ground truth reference. However, it does not mention the prerequisite of having an active slice or provide the link to learn about activating a slice, which is included in the ground truth. Therefore, it is not entirely complete". Score: 0.25.

This example illustrates how human and LLM quality analyses may differ. According to human evaluation, when prompted with "Can you provide an example of attaching a device to a slice using an SDK?", GPT-4's response in Figure 7 did not contain code or text errors and was more similar to the ground truth content in Figure 6, so no points were deducted. However, according to the LLM evaluation, mentioning that it is necessary to have an active slice before attaching a device is considered important, and points were deducted from its evaluation. Perhaps, a human might consider that it is not necessary to repeat information since the active slice requirement for device attachment is mentioned upon slice creation and previous pages in the documentation.

5 Discussion

As seen in the (4.2), human evaluation is still somehow correlated with deterministic metrics, even though not strongly, while LLM evaluation is peripheral, only associating with the human one. Overall, the deterministic metrics had higher correlations amongst themselves, mostly likely due to the token overlap between ground truth and generated content. It is important to remember that "hallucinated" responses can also have high token overlap without necessarily generating more accurate responses relevant to a real use-case scenario.

On the other hand, human and LLM quality analyses seem more closely related to each other, even when using different frameworks to evaluate quality. This shows that an increase in the human evaluation variable relates to an increase in the LLM one. Additionally, the linear regression model plots seemed to reflect similarities in model preference by both human and LLM judgment. However, as the distribution of human quality evaluation scores had several peaks, it could indicate that the human evaluation had a more varied score distribution, due to the detailed error typology followed by the quality analysis type, while the LLM analysis mostly considered correctness and relevance as the main reference. Furthermore, as illustrated by the qualitative example (4.3), the assessments of a human evaluator and the LLM do not always correspond to each other.

6 Conclusion and future work

This paper analyzed technical documentation content generated by three different LLMs using RAG technology and compared the responses to product documentation. Different types of analyses were done: a qualitative analysis using DQF-MQM error typology by a technical writer, an automatic LLM correctness assessment, and a deterministic evaluation using BLEU, ROUGE, and token overlap. The evaluation scores were contrasted and compared using Network Analysis, linear regression models, and a histogram with score distribution. Deterministic metrics have strong relationships with each other, while human analysis correlates moderately with LLM analysis and weakly with deterministic metrics.

The DQF-MQM translation quality evaluation framework was found to be a useful model also for the evaluation of technical writing content and shows potential for improving evaluation methods in the generative AI field. The current study is limited by involving only one technical writer as an evaluator, but the approach can provide a basis for further studies involving multiple evaluators. Future work could also include integrating such qualitative evaluation frameworks to evaluation packages like Continuous-eval for a more granular approach to evaluating LLM output.

Overall, the evaluation results indicate that LLMs are not yet ready as producers of novel content or standalone solutions for technical writing content. Future research is likely to continue exploring methods for enhancing chatbot responses for technical documentation purposes through RAG or different prompting techniques such as chain-of-thought (Wei et al., 2024). Developing novel and reliable quality evaluation methods is therefore also an essential challenge for positive advancements in this area.

7 Limitations

The dataset is relatively small with a total of 72 responses analyzed. However, the responses were relatively long with several of them comprising whole Markdown pages. The evaluations, though subjective with only one evaluator, are informed by their expertise as a professional technical writer familiar with the product. Additionally, using a proprietary RAG chatbot technology limited the possibility of providing a more detailed technical description here.

Acknowledgments

We thank Nokia, particularly the Network as Code Team and chatbot developers, for authorizing data usage for this research.

References

Sheila Castilho, Stephen Doherty, Federico Gaspari, and Joss Moorkens. 2018. Approaches to Human and Machine Translation Quality Assessment. In Joss Moorkens, Sheila Castilho, Federico Gaspari, and Stephen Doherty, editors, *Translation Quality Assessment: From Principles to Practice*, pages 9–38. Springer International Publishing, Cham.

DAIR.AI. Basics of prompting [online]. n.d. Prompt Engineering Guide.

Sacha Epskamp, Denny Borsboom, and Eiko I. Fried. 2018. Estimating psychological networks and their accuracy: A tutorial paper. *Behavior Research Methods*, 50:195–212.

Mikhail Evtikhiev, Egor Bogomolov, Yaroslav Sokolov, and Timofey Bryksin. 2023. Out of the BLEU: How should we assess quality of the Code Generation models? *Journal of Systems and Software*, 203:111741.

Patrick Fernandes, Daniel Deutsch, Mara Finkelstein, Parker Riley, André Martins, Graham Neubig, Ankush Garg, Jonathan Clark, Markus Freitag, and Orhan Firat. 2023. The devil is in the errors: Leveraging large language models for fine-grained machine translation evaluation. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1066–1083. Association for Computational Linguistics.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chikiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George Foster, Alon Lavie, and André F. T. Martins. 2022. 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*, pages 46–68. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv*.

Tom Kocmi and Christian Federmann. 2023a. GEMBA-MQM: Detecting translation quality error spans with GPT-4. In *Proceedings of the Eighth Conference on Machine Translation*, pages 768–775. Association for Computational Linguistics.

Tom Kocmi and Christian Federmann. 2023b. Large language models are state-of-the-art evaluators of translation quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*, pages 193–203. European Association for Machine Translation.

LangChain. Langchain documentation [online]. 2023. GitHub Repository.

Daniel Lüdecke. 2024. *sjPlot: Data Visualization for Statistics in Social Science*. R package version 2.8.16.

Mistral AI. Announcing Mistral 7B: The best 7B model to date, Apache 2.0 [online]. 2023.

Alexandre Nikolaev and Neil Bermel. 2022. Explaining uncertainty and defectivity of inflectional paradigms. *Cognitive Linguistics*, 33(3):585–621.

Nokia. Network as Code portal documentation [online]. 2024. Nokia Network as Code Portal.

OpenAI. Models [online]. 2023. OpenAI Documentation.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. 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. Association for Computational Linguistics.

R Core Team. 2024. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Relari. Open-source evaluation framework and synthetic data generation pipeline [online]. 2023. Continuous Eval.

Karen de Souza. 2024. How much can AI assist in the generation of technical documentation research on AI as a support for technical writers. Master's thesis, University of Eastern Finland.

James Swarts. 2018. *Wicked, Incomplete, and Uncertain: User Support in the Wild and the Role of Technical Communication*. Utah State University Press, Logan.

TAUS. Error annotation based on DQF-MQM [online]. n.d. TAUS, The Language Data Network.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, page 14. Curran Associates Inc.

Hadley Wickham. 2016a. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Hadley Wickham. 2016b. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. 2019. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

Yi Zhang and Pasquale Antonante. A practical guide to RAG pipeline evaluation (Part 1: Retrieval) [online]. 2023.

A Appendices

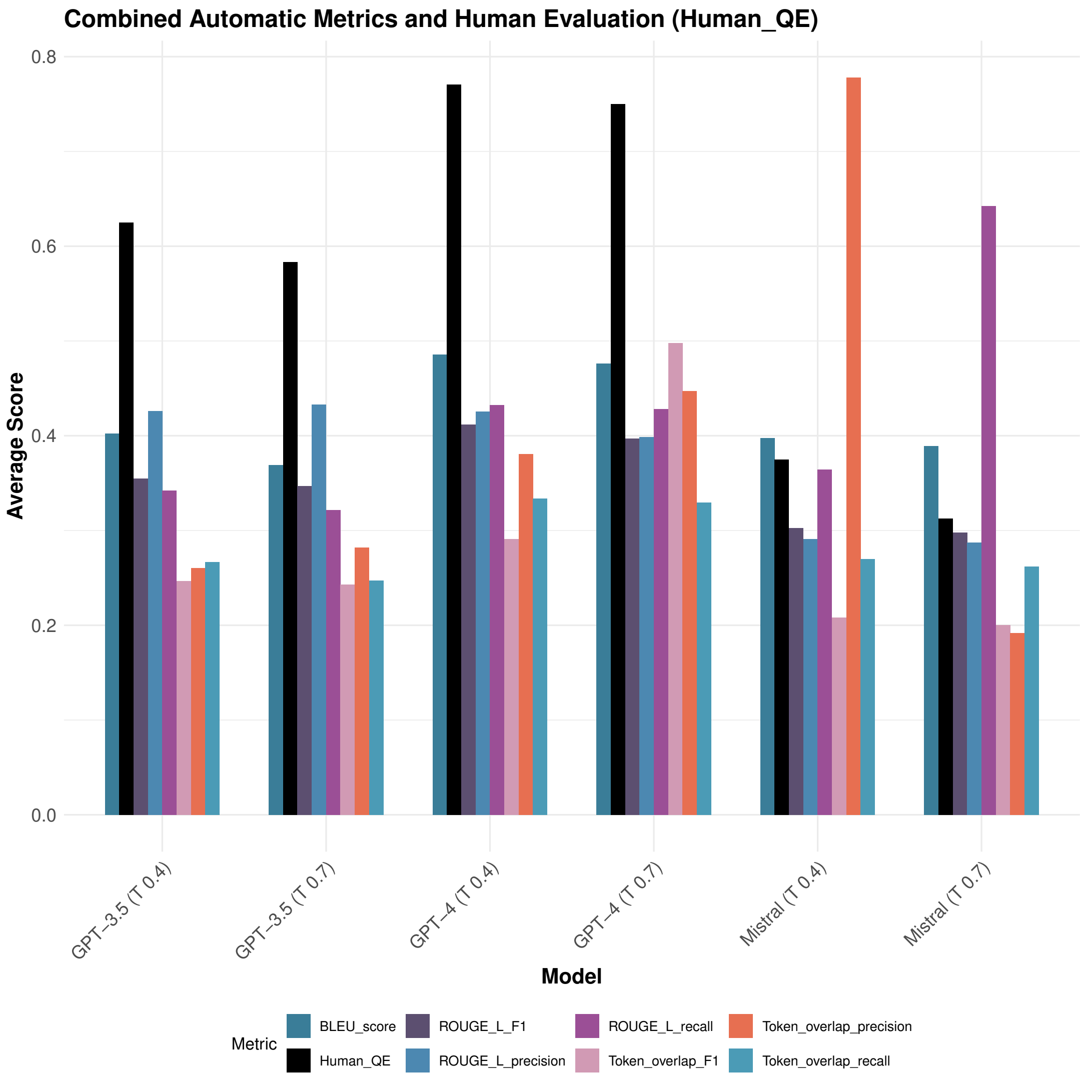


Figure 8: Human QE and deterministic averages

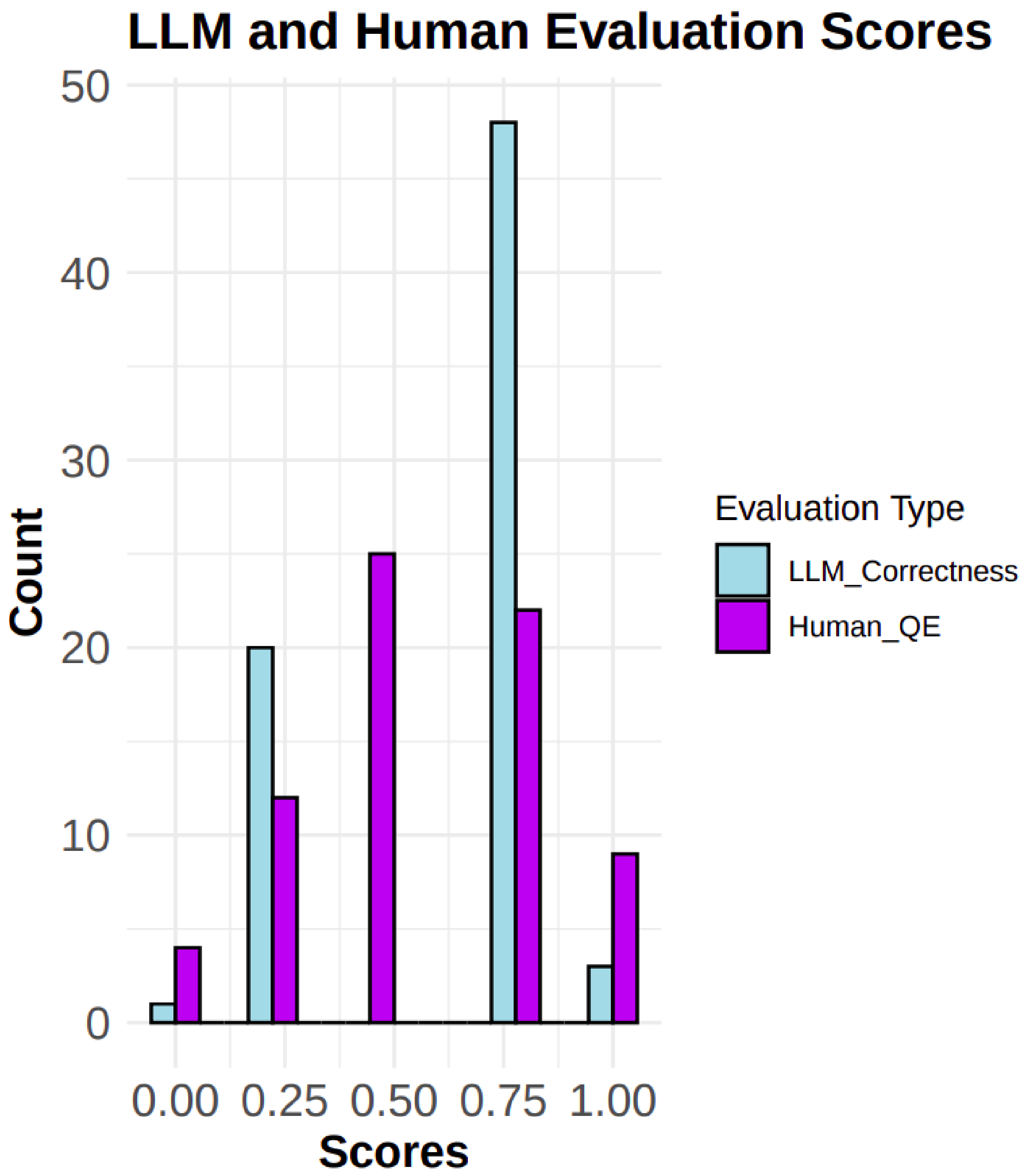


Figure 9: LLM and human scores distribution

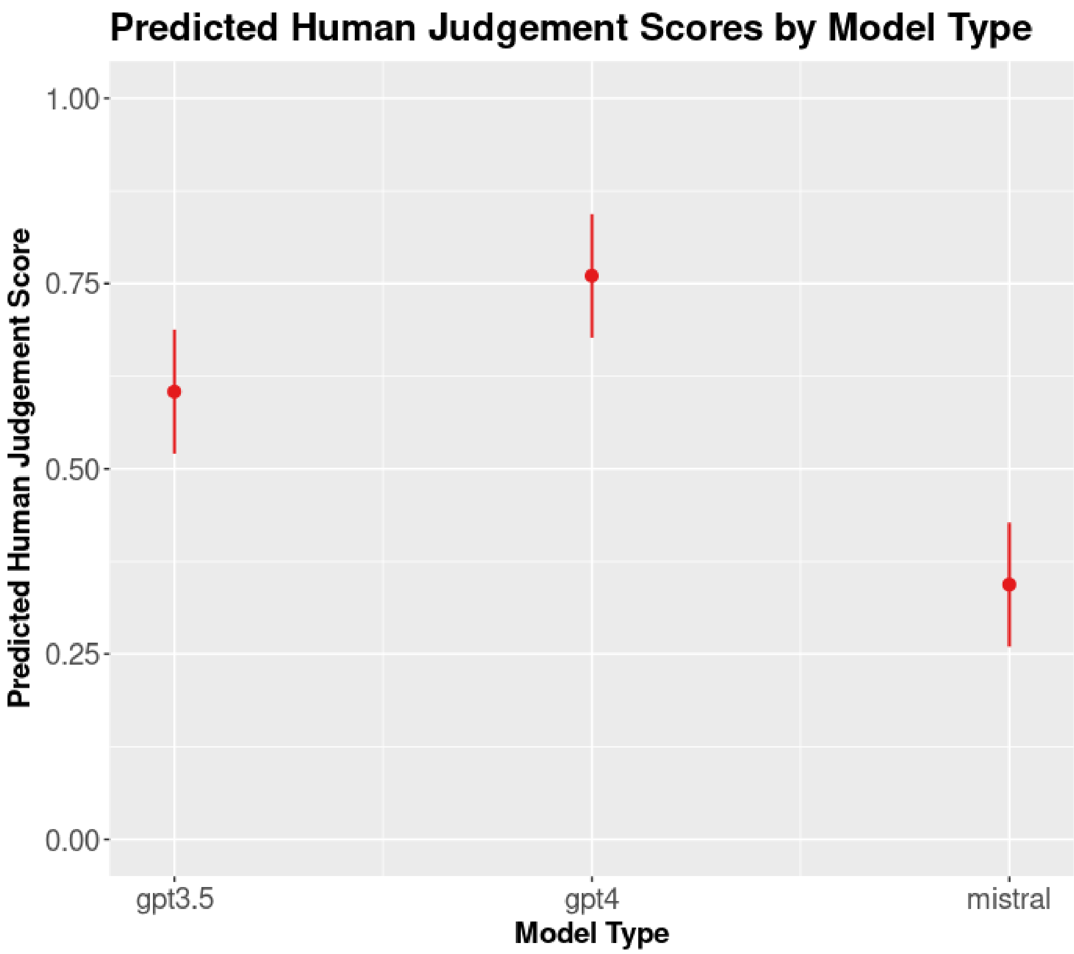


Figure 10: Human analysis linear regression model

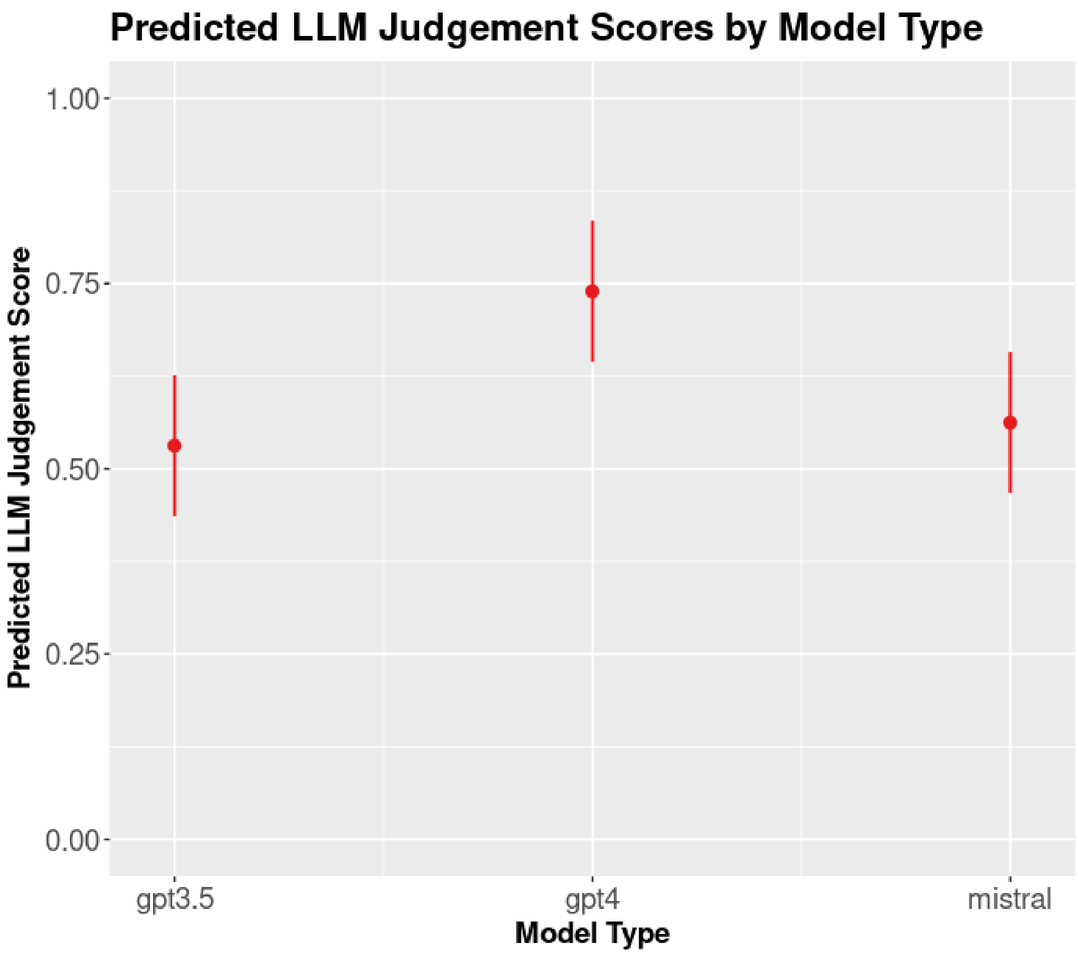


Figure 11: LLM analysis linear regression model

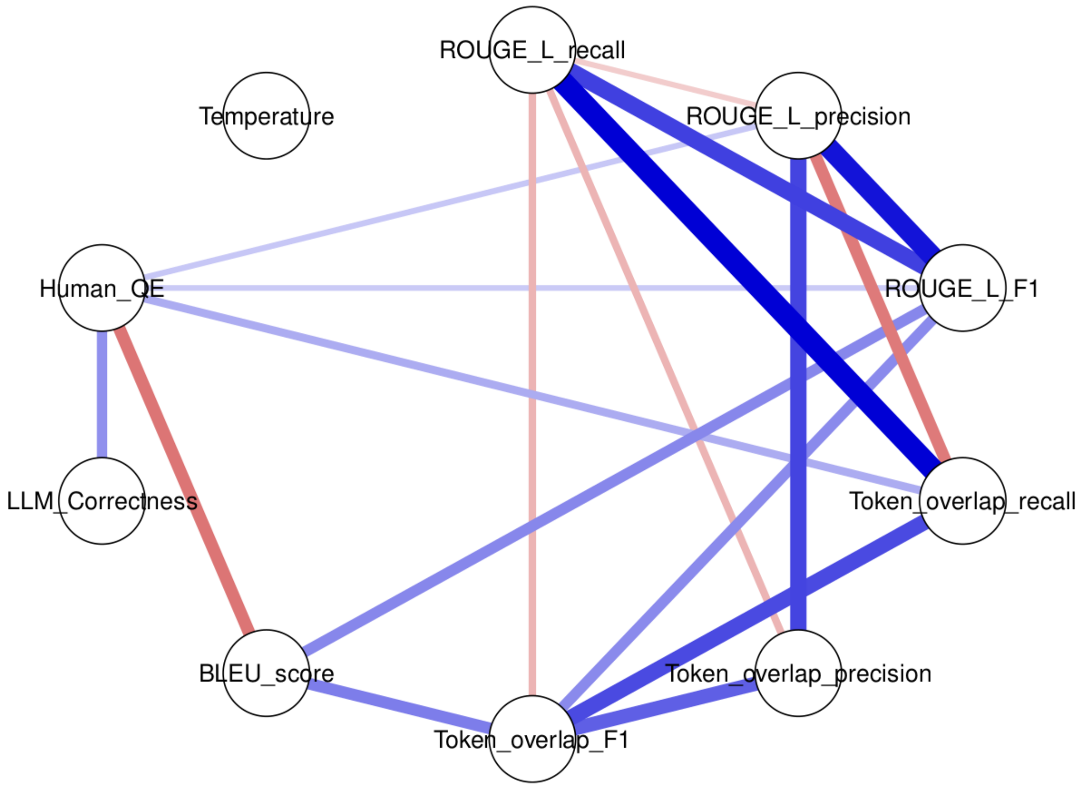


Figure 12: Correlation Network Analysis

```
Device attachment presupposes a slice already exists and that it is activ
Learn how to activate a slice
[here](/docs/slicing/slice-activate-deactivate#activating-a-slice).
So, after creating and activating the desired slice
according to the steps in the [previous page](/docs/slicing/slice-creatio
simply use the `slice.attach()` method to attach a device.

```python
from network_as_code.client import NetworkAsCodeClient

from network_as_code.models.device import DeviceIpv4Addr

from network_as_code.models.slice import(
    NetworkIdentifier,
    Slice,
    SliceInfo,
    AreaOfService,
    Point,
    Throughput
)

device = client.devices.get(...)

slice = client.slices.create(...)

slice.attach(
    device,
    "https://example.org/notify",
    notification_auth_token="replace-with-your-auth-token"
)
```
```

Figure 13: Ground truth context reference

Certainly! Below is an example of how to attach a device to a network slice using the Network as Code SDK. This example assumes that you have already created and activated a slice, and that you have a Network-as-Code client and a device object ready.

```python
from network_as_code.client import NetworkAsCodeClient
from network_as_code.models.device import DeviceIpv4Addr
from network_as_code.models.slice import(
    NetworkIdentifier,
    Slice,
    SliceInfo,
    AreaOfService,
    Point,
    Throughput
)

# Initialize the Network-as-Code client with your token
client = NetworkAsCodeClient(token="<your-application-key-here>")

# Get the device object using its identifier
device = client.devices.get("<device-identifier>")

# Get the slice object using its name or identifier
slice = client.slices.get("<slice-name>")

# Attach the device to the slice
slice.attach(
    device,
    "https://example.org/notify",
    notification_auth_token="replace-with-your-auth-token"
)
```

Figure 14: GPT-4 response

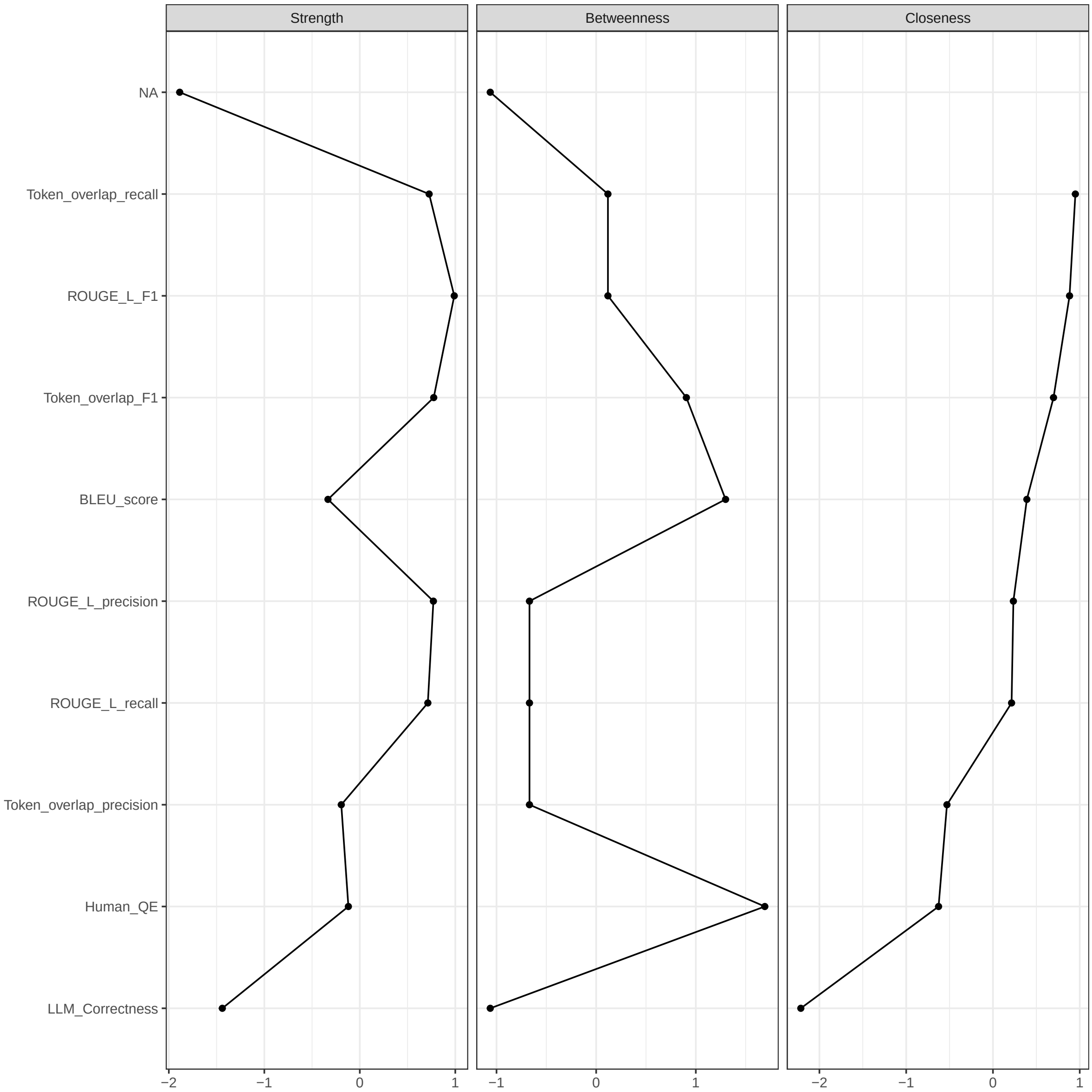


Figure 15: Closeness centrality index

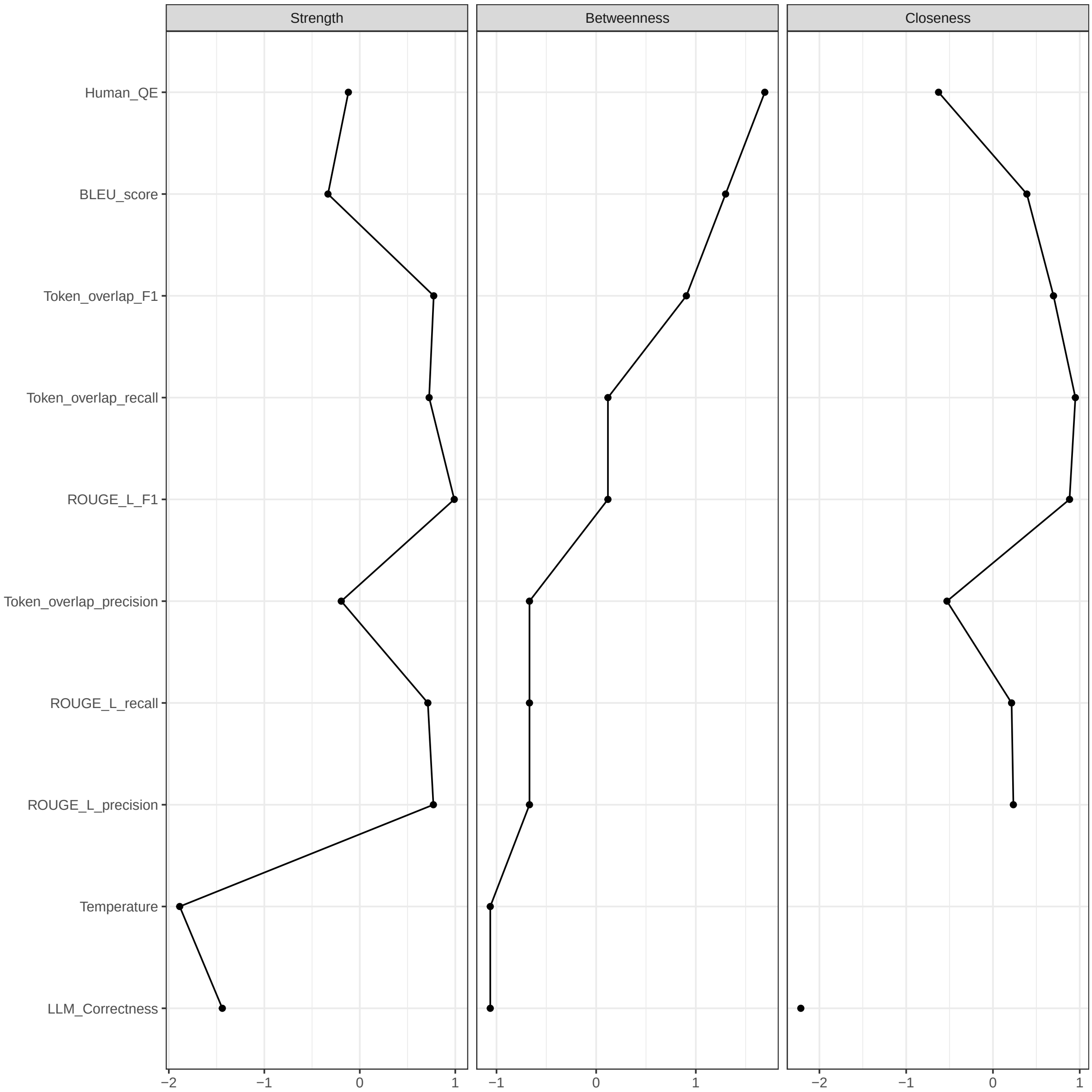


Figure 16: Betweenness centrality index

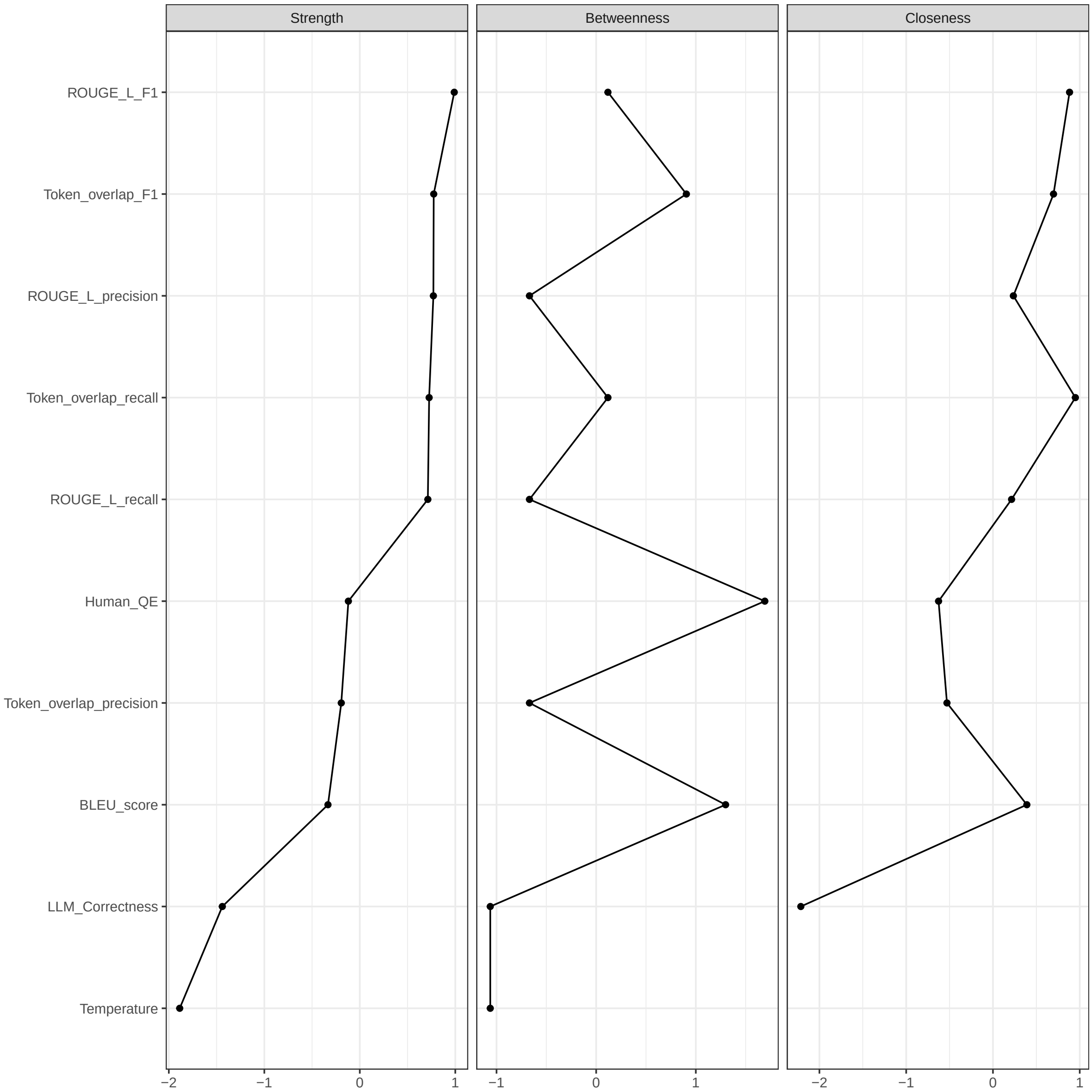


Figure 17: Strength centrality index