Evaluating Llama-2 and GPT models on the Grammatical Errors Correction Task

Hung Hoang - hunghoang26@uwo.ca

I. Introduction

Grammatical errors correction (GEC) is a common task that is carried out on a daily basis by non-native and native speakers alike. For foreign language learners, it plays an even more special role during their learning process as this helps them identify, fix and learn from their writing mistakes.

In this project, I evaluated the GEC performance of two of the most recent and well-known Large Language Models (LLMs): Llama-2 models from Meta and GPT models from OpenAI. The evaluation was carried out following the protocol of the Building Educational Application (BEA) 2019 Shared Task on GEC [1, 2] for the English language.

The results showed that, when evaluated using this framework, larger-version LLMs did not perform better. This surprising result, after analysis, has revealed that in the context of LLMs, the current GEC evaluation approach is no longer adequate.

II. Background on BEA 2019 Shared Task and Evaluation

The BEA 2019 Shared Task on GEC helped push evaluation of GEC systems forward by making 2 concrete contributions: (1) providing common labeled datasets for training and evaluation and (2) incorporating the evaluation toolkit called ERRANT [3].

Datasets

In this project, I made use of off-the-shell LLMs and as such, did not make use of the datasets described below for training. There are, therefore, only described briefly below.

The BEA 2019 Shared Task introduced 2 new labeled datasets for training and standardized some previously published GEC datasets by running the ERRANT Toolkit over them. More details of these other datasets are provided in the Shared Task website [1] while the ERRANT Toolkit is described in the next section.

Besides the Training set, this Shared Task used a Test set of 4,447 sentences, which was also used for its ongoing Leaderboard [4]. The gold labels for the Test set were not made available and hence, the only way to measure the system's performance over this data set is to submit correction outputs to the Leaderboard evaluation system.

The ERRANT Toolkit

The introduction of ERRANT (ERRor ANnotation Toolkit) [3] as a common evaluation metric enabled quick comparison among different GEC systems.

This toolkit requires 2 files as inputs: one containing original sentences and one containing corrected sentences in the same order. This toolkit then automatically produces a file, commonly referred to by the author as an "M2" file, listing the corrections made for each sentence, and for each correction, automatically identifying the error category. In total, there are 25 main error categories recognized by ERRANT, using about 50 coded rules.

To make it clearer, I go through a specific example and explain the various labels and outputs by ERRANT.

- Input: *Travelling through car is a convenient mode of transport , but most of the population can not afford it .*
- Corrected (by GPT 3.5): *Traveling by car is a convenient mode of transport , but most of the population can not afford it .*

M2 file produced by ERRANT:
- S Travelling through car is a convenient mode of transport , but most of the population can not afford it .
- A 0 1|||R:VERB:FORM|||Traveling|||REQUIRED|||-NONE-|||0
- A 1 2|||R:PREP|||by|||REQUIRED|||-NONE-|||0

Explanations:
- The first line starts with "S" indicating the original sentence to fix. For ERRANT to work, the input pair (original sentence and corrected one) must have been tokenized using the Spacy library [5], as shown in the sentence pairs above. The general effect of this tokenization is simply introducing a space before each word and punctuation mark. This tokenization is not the same as the more complex tokenization done by LLMs where words may be split into subwords.

- The next lines always start with "A" and contain information about corrections made. Each correction has its own line with the following format:
  A <start index> <end index> |||<R/U/M>: Classified error type>|||<Correct word/phrase>|||REQUIRED|||-NONE-|||<Annotator id>
  - Fields separator is "|||".
  - There are 2 fields that always have the same value, and are there only for legacy purposes: "REQUIRED" and "-NONE-".
  - <start index> and <end index>: refer to the index of the incorrect string in the original sentence. This index uses Python semantics: sentence[start index : end index]. In the first line, this refers to the word "Travelling".
  - The edit type, which can be one of these 3 letters: R, U and M:

        - R: replacement. This means the original string is replaced with a new string.
        - U: unnecessary. This means the original string is deleted.
        - M: missing. This means a new string is inserted at the specified index. (In this case, <start index> would be equal to <end index>).
    - <Classified error type>: one of the error categories recognized by ERRANT; if not recognized, “OTHER” is shown.
    - <Correct word/phrase>: the correct version suggested by the GEC program.
    - <Annotator id>: the id of the annotator, starting with 0. In our example, the corrected version was produced by 1 annotator (our system) with id 0. This field was used to identify which corrections were suggested by which annotator, in the case where a sentence might have been annotated by multiple annotators.

In the above example, GPT3.5 made 2 corrections:

- Changed “Travelling” to “Traveling” (from double “l” to a single “l”). This correction was classified by ERRANT as a “VERB:FORM” type (a correction of verb forms).
- Changed “through” to “by”. This correction was classified by ERRANT as a “PREP” type (a correction of prepositions).

Below is another example produced by Llama-2 70B model:

- Input: *It was built in 79 A.C. by Tito , who was the Roman emperor .*
- Corrected: *It was built in AD 79 by Titus , who was the Roman emperor .*

M2 output produced by ERRANT:

- S It was built in 79 A.C. by Tito , who was the Roman emperor .
- A 4 4|||M:NOUN|||AD|||REQUIRED|||-NONE-|||0
- A 5 6|||U:NOUN|||||REQUIRED|||-NONE-|||0
- A 7 8|||R:NOUN|||Titus|||REQUIRED|||-NONE-|||0

Explanations:

- The first edit is an “M” (missing) edit, which means new tokens were inserted: at index 4, Llama suggested adding “AD”.
- The second edit is an “U” (unnecessary) edit, which means existing tokens were deleted: the token at index 5 (“A.C.”) was deleted.
- The third edit is an “R” (replacement) edit, which means existing tokens were replaced with new tokens. Here, Llama replaced “Tito” with “Titus”.

Evaluation Metric

The metric used for the BEA 2019 Shared Task is $F_{0.5}$ of Precision and Recall.

$$F_\beta = \frac{(1+\beta^2)\times Prec \times Recall}{(\beta^2 \times Prec)+Recall}$$

where:

- Precision: the ratio of correctly identified errors over all errors identified by the system. The list of all errors are given by the provided gold M2 file. A correction is considered correct if both the location of the fixed tokens in the input string and the corrected tokens are both correct, i.e., all <start index>, <end index> and <correct word/phrase> must be all correct. As this takes into consideration the location of the error, this was referred to as “Span-Based Correction” the terminology of the BEA 2019 Shared Task.
- Recall: the percentage of all errors given by the gold M2 file that were correctly identified by the system.
- Beta: in our case is 0.5. A beta of 0.5 makes Precision more important than Recall, as the same percentage increase in Precision, compared with Recall, would cause a much less increase in the value of the denominator.

This Shared Task chose to use $F_{0.5}$ instead of F_1 to assign more weight to Precision than Recall. In educational settings, this makes sense as, for example, a real teacher would rather “overlook” some errors than mistakenly mark a correct phrase as an error.

The Unrestricted Track

The BEA 2019 Shared Task consisted of 3 main tracks:

- Restricted Track: where only training data as described in the “GEC Datasets” Section above can be used to train systems.
- Unrestricted Track: any dataset and software can be used.
- Low-resource Track: only certain corpora could be used for training.

The scope and completion time of this project, if it needs to be classified into one of the above tracks, would belong to the Unrestricted Track. This track has an “open phase” where there is no set deadline and anyone can submit their results over the Test Set for evaluation.

III. Methods

Evaluated Models and Methodology

In this project, I evaluated the (open-sourced) Llama-2 models [6] and (proprietary) GPT models [7], using the BEA 2019 evaluation protocol described above.

In particular, the following specific models are evaluated:

- Llama-2 13B Chat (13B: refers to 13 billion parameters)
- Llama-2 70B Chat
- GPT 3.5 (Turbo)
- GPT 4

- GPT 4 1106 preview (which was made available after OpenAI Dev date on 6/Nov/2023).

All of the above LLMs are generative models (or auto-regressive models) with Transformer [8] as their underlying architecture. They have all been pretrained on enormous datasets. As these models are very recent and complex, a more detailed description of their particular architecture tweaks and pretraining process is beyond the scope of this project and is hence omitted here. This omission does not affect this project's goals.

As these LLMs are known to be able to capture language syntax and semantics, I experimented with evaluating them for this GEC task in a zero-shot manner, i.e., using these pretrained models as is, except that, as described in the following Experiments section, some demonstration examples are used to communicate the desired output format to these LLMs.

The models above are invoked via API calls provided by OpenAI for GPT models [7] and OpenRouter [9] for Llama models. After getting the response, each corrected sentence is then tokenized using Spacy before passed to ERRANT to produce an M2 file containing corrections' annotations. This M2 file is then compared with the gold M2 file to calculate Precision, Recall and $F_{0.5}$.

For the purpose of this project, besides assessing whether GEC is a solved problem, another question of interest is how much the bigger models would outperform smaller ones. In practical applications, smaller models *are* preferred, if they can reach a target performance, as they help reduce both inference time and cost.

Our Evaluation Dataset

As the outputs of these LLMs are produced via paid APIs provided by third-parties, experimenting over this entire Test set would not only be more costly but also take substantially longer time to complete. But more importantly, as the Test set's labels are not provided, it would not be possible to "debug" our results.

As such, I looked at the provided 4,384-sentence Dev set, whose labels are available. To reduce experiments running time and cost, I took out from the Dev set a random 400-sentence subset to form my evaluation set. This 400-sentence evaluation dataset is stratified based on the length of the sentences as follows: first sort the Dev set by sentences' lengths and then partition them into 4 equal sets. From each set, randomly take 100 sentences to form our 400-sentence evaluation data set.

The rationale for stratifying based on sentences' length is that, from both experience and common sense, the length of a sentence has a high correlation with the number of grammatical errors. As such, stratifying our evaluation data set based on length will make our results more closely match the results of evaluated on the entire Dev set.

IV. Experiments and Results

As Llama-2 models are open-sourced with lower API costs, I experimented first with these models. In all of the experiments, sentences are put together in a batch instead of being sent individually as batching would help reduce response latency and cost as we only need to send the prompt for each batch instead of each sentence.

For all models, the system message, which was sent together with each prompt, was: “You are a dedicated English teacher.”

Llama-2 13B Model

Below is the best-performing prompt for the Llama-2 13B model.

Prompt used for Llama-2 13B model
Please fix all grammar errors in the following sentences. Do not change the words if they are not wrong: be conservative and make changes only when necessary. Each sentence has the format: {sentence number}: {content}. Fix the content of each sentence individually and return the one best corrected sentence. Do not return anything else besides the corrected version. 1: As well , the fact that so many people (especially in the US) have television sets means that everybody (well , at least everybody who watches) receives the same inflow of information and ideas . 2: It is based on a survey of the young people from Whitehall College as well as my own opinion as a permanent resident in the area . …

This prompt was arrived at after some experimenting to deal with a prominent problem with this 13B model: it did not fix all the sentences in the input, e.g., returning only 15 fixed sentences for an input batch of 20 sentences.

In addition, Llama 13B’s outputs were also not consistent in format, causing problems in parsing its responses. As can be seen in the above prompt, although I have tried to explicitly add “*Do not return or say anything else besides the corrected version.*” in the prompt, its outputs’ format was still not consistent. Below are 2 examples of this behavior:

1: I'm looking forward to seeing you.
Corrected: I'm looking forward to seeing you.

2: In Malaysia, the weather is slightly warmer than there. (Should be "the weather is slightly warmer than here.")

In the first example, it introduced, on its own, the prefix “Corrected: “ while in the second example, it repeated the input sentence and introduced the corrected fragment in a bracket.

Eventually, to deal with its unexpected outputs, I had to fix them manually by either making additional API calls for missing sentences or to format its outputs to the expected form. In general, its unexpected outputs made this model less useful in practice.

Below is its performance on our 400-sentence evaluation dataset:

TP	FP	FN	Prec	Rec	F0.5
276	686	380	0.2869	0.4207	0.3064

Table 1: Result of Llama-2 13B model

The Precision was rather low so I proceeded to experiment with the larger 70B version to see if this larger version could achieve better results.

Llama-2 70B Model

Below is the best performing prompt of Llama-70B using demonstrations:

```
Prompt used for Llama-2 70B model

Please fix all grammar errors in the following sentences, some of them may contain only short phrases. Do not change the words if they are not wrong: be conservative and make changes only when necessary. For each sentence, return one best corrected version, starting your sentence with "Fixed: ". Do not repeat the example sentences.

Follow these 2 examples:
1. It 's difficult answer at the question " what are you going to do in the future ? " if the only one who has to know it is in two minds .
1. Fixed: It's difficult to answer the question "what are you going to do in the future?" when the only one who has to know it is in two minds.

2. When I was younger I used to say that I wanted to be a teacher , a saleswoman and even a butcher .. I do n't know why .
2. Fixed: When I was younger, I used to say that I wanted to be a teacher, a saleswoman, and even a butcher. I don't know why.

1. As well , the fact that so many people ( especially in the US ) have television sets means that everybody ( well , at least everybody who watches ) receives the same inflow of information and ideas .
```

```
2. It is based on a survey of the young people from Whitehall College as well as my own
opinion as a permanent resident in the area .

…
```

A marked difference of this 70B compared to its smaller sibling is that it followed instructions and demonstrations much better, except for a few cases of inconsistent format. For the smaller 13B model, when experimented using the above prompt (with demonstrations), its output format was still inconsistent.

Below is the evaluation result for this 70B model.

TP	FP	FN	Prec	Rec	F0.5
298	728	358	0.2904	0.4543	0.313

Table 2: Result of Llama-2 70B model

It was really surprising to me that this 70B version, despite its much better instruction following capability, did not perform better than its 13B sibling on Precision. On Recall, it improved by just 3%.

Next, I moved on to the GPT models to see if they could do better.

GPT 3.5 (Turbo) Model

I experimented with GPT3.5 using exactly the same as the prompt I used above with the Llama-2 70B model (with demonstrations). Below is its evaluation result:

TP	FP	FN	Prec	Rec	F0.5
325	496	331	0.3959	0.4954	0.4124

Table 3: Result of GPT 3.5 model

Compared with the Llama-2 70B model, GPT3.5 improved Precision by 10.5% and Recall by 4%.

GPT3.5 also followed instructions perfectly (no manual edits needed), and a much larger batch size could be used, until reaching the max context window of 4K for this model, at a batch size of about 60 sentences. In contrast, for Llama-70B, I could only use a batch size of 15 sentences, because increasing the batch size more than this limit will cause it to ignore subsequent input sentences.

Before experimenting with GPT4, I tried to add 2 no-error sentences to the existing 2-example demonstration to experiment with whether this helped reduce false positive cases by explicitly telling the model that it does not always need to produce a fix if the input sentence is already correct. The evaluation result nevertheless did not show any material improvements.

If GPT3.5 could increase the Precision by 10%, my hope was that GPT4 would add, at least, another 10% - 20% in Precision.

GPT 4 Models

For GPT4 models, I used the same prompt that was used for the GPT3.5 model above.

TP	FP	FN	Prec	Rec	F0.5
294	628	362	0.3189	0.4482	0.3384

Table 4: Result of GPT 4 model

TP	FP	FN	Prec	Rec	F0.5
337	610	319	0.3559	0.5137	0.3792

Table 5: Result of GPT 4-1106 Preview model

GPT4-1106 Preview performed better than GPT-4 by 4% in F0.5, but both performed worse than GPT3.5 in Precision (0.3959), and only less than 2% improvement in Recall from the 1106-Preview model.

As this result totally contradicts our common knowledge that GPT4 is much superior to GPT3.5, this warranted some investigation on its unexpected poor performance. The results of my manual analysis of its outputs are discussed in the next section.

V. Analysis and Discussion

To discover the reason for GPT4's poor performance and LLMs' disappointingly low performance in general, I manually looked at their outputs and compared them with the gold labels.

It turned out that LLMs made substantially more corrections than what were covered by the gold labels, i.e., producing many false positives (from the viewpoint of the gold labels).

These corrections may or may not be necessary, but they are by no means incorrect. Below I give 2 concrete examples of this behavior: one from Llama-70B and one from GPT4.

An example output from Llama-2 70B

Input: *Granted , it is the person who " told " the computer what to do that truly made the error ; however if that person knew what he should do , or better stated " was able to do with a computer " , he would not have erroneous answers .*

Output: *Granted , it is the person who told the computer what to do that truly made the error ; however , if that person knew what they should do , or better stated , " was able to do with a computer , " they would not have received erroneous answers .*
Output M2:

- A 7 8|||U:PUNCT|||||REQUIRED|||-NONE-|||0
- A 9 10|||U:PUNCT|||||REQUIRED|||-NONE-|||0
- A 22 22|||M:PUNCT|||,|||REQUIRED|||-NONE-|||0
- A 27 28|||R:PRON|||they|||REQUIRED|||-NONE-|||0
- A 34 34|||M:PUNCT|||,|||REQUIRED|||-NONE-|||0
- A 42 44|||R:WO|||, "|||REQUIRED|||-NONE-|||0
- A 44 45|||R:PRON|||they|||REQUIRED|||-NONE-|||0
- A 48 48|||M:VERB|||received|||REQUIRED|||-NONE-|||0

Gold output: *Granted , it is the person who " told " the computer what to do that truly made the error . However , if that person knew what he should do , or better stated " was able to do with a computer " , he would not have erroneous answers .*
Gold M2:

- A 20 22|||R:OTHER|||. However ,|||REQUIRED|||-NONE-|||0

From this example, we could observe the following:

- Llama-70B kept "; " as the connector between 2 clauses while the gold label wanted to make them 2 separate sentences. As there is only 1 labeled fix, and the model missed it, recall for this case is 0%.
- The model made many more corrections not covered by labeled data, such as 2 fixes about changing "he" to "they" for pronoun reference, which I think is correct. The model also changed "would not have erroneous answers" to "would not have received erroneous answers" (adding "received"), which is again not covered by labeled data. This makes the model's precision for this example 0%.

This example demonstrates that while the model's corrections are decent, it received 0% for both recall and precision due to low coverage of labeled data. LLMs in general tend to make more number of corrections and the changes they make tend to be "out of the box" kinds of changes. This observation helps explain their surprisingly low precision and recall, despite the fact that their outputs are, arguably, correct.

An example output from GPT4

Input: *Before training i ran 1 - 2 kilometres and my legs get a power and my body gets an energy .*

Output: *Before training , I ran 1 - 2 kilometres giving power to my legs and energy to my body .*
Output M2:
- A 2 3|||M:PUNCT|||, I|||REQUIRED|||-NONE-|||0
- A 8 9|||R:OTHER|||giving power to|||REQUIRED|||-NONE-|||0
- A 11 14|||U:OTHER||||||REQUIRED|||-NONE-|||0
- A 15 15|||M:OTHER|||energy to|||REQUIRED|||-NONE-|||0
- A 17 20|||U:OTHER||||||REQUIRED|||-NONE-|||0

Gold output: *Before training , I run 1 - 2 kilometres and my legs get power and my body gets energy .*
Gold M2:
- A 2 2|||M:PUNCT|||,|||REQUIRED|||-NONE-|||0
- A 2 3|||R:ORTH|||I|||REQUIRED|||-NONE-|||0
- A 3 4|||R:VERB:TENSE|||run|||REQUIRED|||-NONE-|||0
- A 12 13|||U:DET||||||REQUIRED|||-NONE-|||0
- A 18 19|||U:DET||||||REQUIRED|||-NONE-|||0

Technically, all corrections made by the model are deemed incorrect according to gold labels, and this means 0% for both precision and recall for this input.

Semantically, the differences in the model's corrections and those of the labeled data are:
- The model kept the past tense "ran", while the labeled corrections changed it to the present tense: "run". This change is, in my opinion, optional.
- The labeled corrections tried to keep as much of the original sentence as possible and made minimal changes, in both cases dropping the article in front, to make the sentence syntactically correct:
  - "my legs get a power" → "my legs get power"
  - "my body gets an energy" → "my body gets energy"
- Meanwhile, the model rewrote the phrase, still using keywords such as "power" and turned it into: "giving power to my legs and energy to my body". I consider this a more fluid expression.

Follow-up experiments

From the above observations, I made 2 follow-up experiments:

Experiment 1:
From the above observation, I tried a final tweak to the prompt to explicitly tell the LLMs:

- Focus on fixing existing words rather than introducing new words. (This is another attempt to reduce false positives)
- Not to fix British English spelling
- Not to correct factual errors

Nevertheless, this new prompt did not produce any material improvements in results.

Experiment 2:
Most LLMs generate new words via sampling based on probabilities assigned to each possible token in the vocabulary set. There is a parameter called "temperature" that can influence this process as it affects the probability assigned to each word at the softmax layer. Decreasing the temperature will make the probability distribution become sharper and hence, making the sampling results become more deterministic.

To make the corrections more deterministic, which may help reduce the high false positive rates of these LLMs, I experimented with setting the temperature value to 0, which was the lowest possible value for both GPT and Llama2 models. In previous experiments, I used the default values of this parameter, which was 1 for GPT models (with the possible range being [0, 2]) and 0.8 for Llama2 models (with range [0, 1]).

Table 6 below shows results when temperature is set to zero. For easy comparison, I also include previous best results of each model.

Model	Previous results			Results with temperature = 0		
	Prec	Rec	F0.5	Prec	Rec	F0.5
Llam2 70B	0.2904	0.4543	0.313	0.1798	0.4253	0.2032
GPT 3.5	0.3959	0.4954	0.4124	**0.4503**	0.4909	**0.4579**
GPT 4	0.3189	0.4482	0.3384	**0.4515**	**0.4893**	**0.4586**
GPT 4 Preview	0.3559	0.5137	0.3792	**0.4226**	0.4954	**0.4354**

Table 6: Results with default temperature and temperature set to 0.
Bolded numbers indicate improvements.

The above results indicated improvement for all GPT models, with most of the gains coming from increased precision score, as expected. And we can also observe that this improvement in precision score did not come at the expense of recall. In this new setting, F0.5 of GPT 3.5 and GPT4 are equal.

It is surprising, however, that for Llama2 70B, both precision and recall deteriorated with temperature set to 0.

VI. Conclusions and Future work

Advances in deep learning research, including the introduction of the Transformer architecture, has made possible pretraining of LLMs on a massive amount of data. This project explored whether GEC as a problem has been solved by recent LLMs, which has shown remarkable ability in capturing linguistic property of language.

This project's results indicated that when using Bea 2019 Shared Task's labeled data for evaluation, LLMs did not perform well in general, and with the surprising result that bigger models (Llama-2 70B, GPT4 and GPT4-Preview) did not perform better than GPT3.5, which is supposedly a much smaller model.

However, my manual analysis of these models' outputs showed that LLMs' corrections are more fluid and diverse than what might have been expected by annotators of current labeled data. As such, many of the corrections of LLMs were not covered in the labeled data, which lead to both a low Precision, due to high false positive cases, and a low Recall due to LLMs' different ways of making corrections.

The low performance of LLMs means that GEC is not a solved problem by this evaluation's standard. Although it is quite reasonable, in my opinion, to suppose that GEC is a solved problem by LLMs, we currently can not make this conclusion using the current evaluation method. In order to benchmark performance of different GEC systems that are based on LLMs, a new evaluation approach seems warranted.

A direct solution to this evaluation challenge is to add more labeled corrections to the gold data, but this approach seems costly and does not scale to new evaluation datasets. Another problem with adding more labeled corrections is that we may need to introduce some kind of semantics to the M2 file to group labeled corrections together, as certain corrections only make sense if combined with certain other corrections for a given sentence.

Another possible direction is to view GEC as a subproblem under a more general problem category such as text rewriting or text summarization and benefit from existing evaluation metrics for those categories.

References

[1] Building Educational Applications 2019 Shared Task: Grammatical Error Correction. https://www.cl.cam.ac.uk/research/nl/bea2019st/

[2] Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (2019, August). The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 52-75).

[3] Bryant, C. J., Felice, M., & Briscoe, E. (2017, July). Automatic annotation and evaluation of error types for grammatical error correction. Association for Computational Linguistics.

[4] BEA 2019 Shared Task - Grammatical Error Correction - All Tracks. https://codalab.lisn.upsaclay.fr/competitions/4057#results

[5] Honnibal, M., & Montani, I. (2017). *spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing*.

[6] Meta. Llama. https://ai.meta.com/llama/

[7] OpenAI. Models. https://platform.openai.com/docs/models

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

[9] OpenRouter. A unified interface for LLMs. https://openrouter.ai/docs