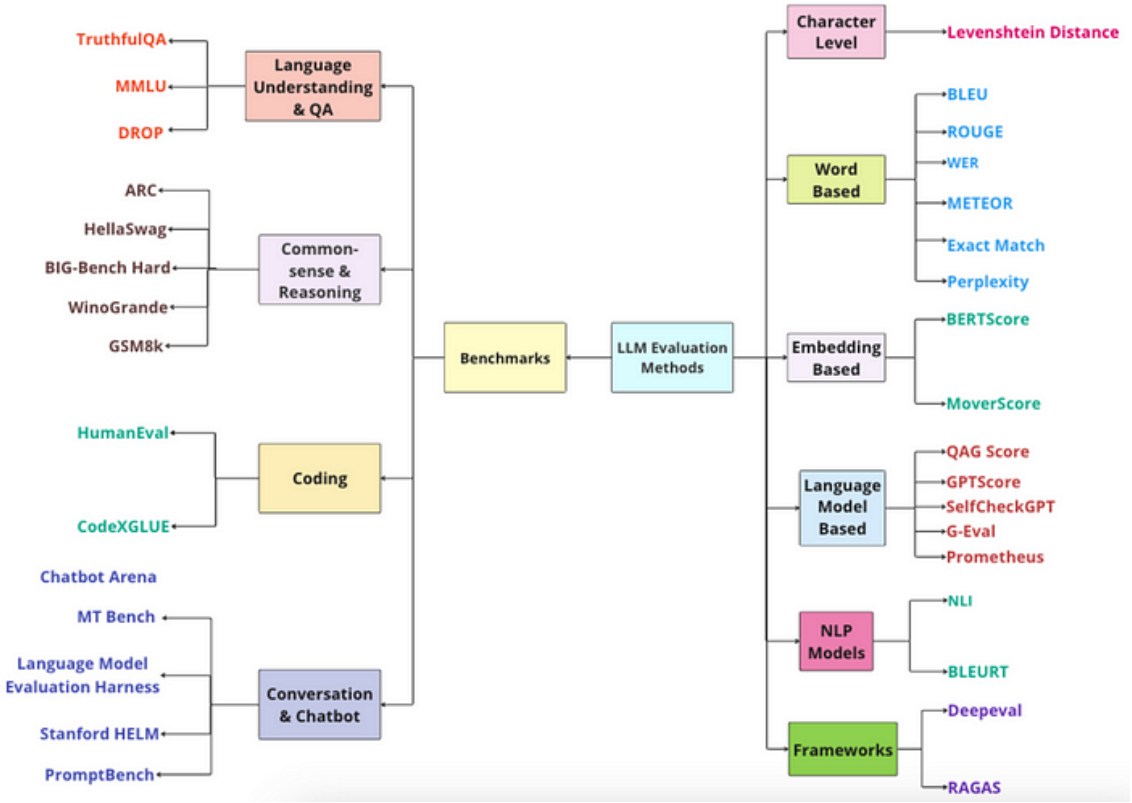
The purpose of this document is to compile and summarize various methods for evaluating language models, including the evaluation of the RAG application.



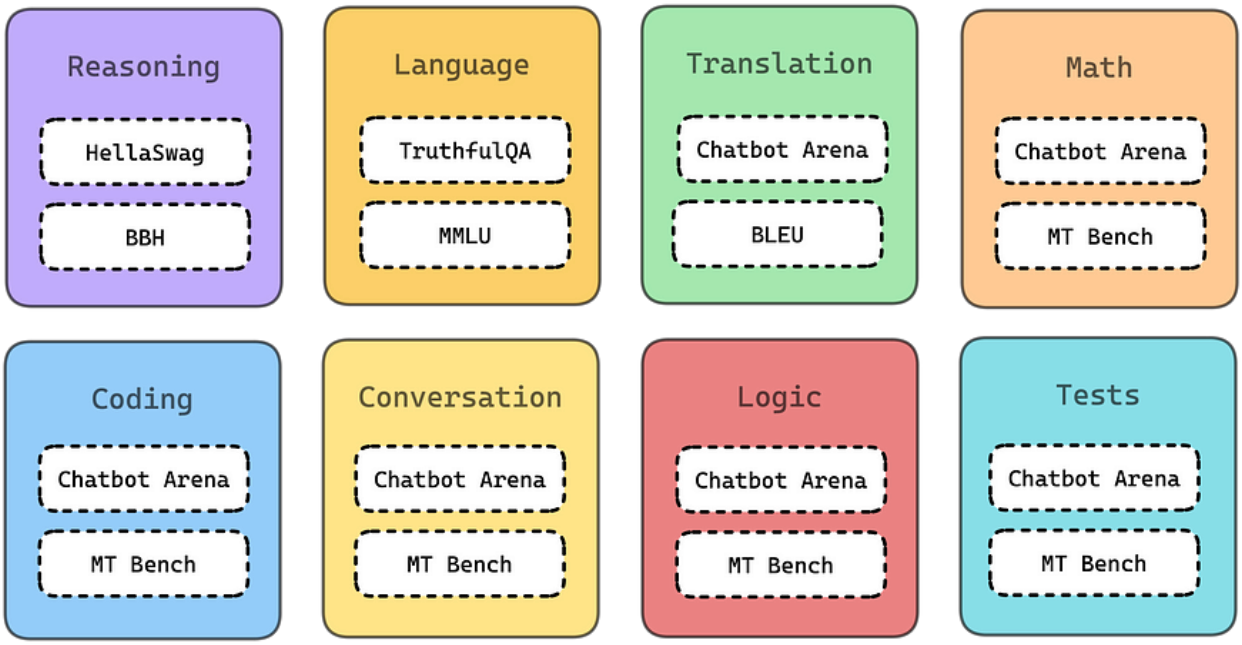
LLM Benchmarking Vs. Evaluation

Benchmarking is all about standardized testing. It involves using predefined datasets and metrics to assess an LLM's performance on specific tasks. Think of it like giving a language model a battery of tests in reading, writing, and math.

Evaluation on the other hand, has a broader scope. It goes beyond just running tests and involves a more holistic assessment of the LLM's capabilities.

1.**LLM Benchmarking**

LLM benchmarks are a set of standardized tests designed to evaluate the performance of LLMs on various skills, such as reasoning and comprehension, and utilize specific scorers or metrics to measure these abilities.



1.1 Language understanding

1.1.1 TruthfulQA

A benchmark consisting of 817 questions across 38 categories related to health, law, finance, and governance. The purpose is to enable the model to avoid or prevent questions that might lead to misunderstandings.

1.1.2 MMLU (Massive multitask language understanding)

A dataset comprising over 57 tasks, including calculations, history, law, and more, in a multiple-choice format. Its purpose is to evaluate pre-trained models by focusing on few-shot and zero-shot performance.

1.2 Common-sense and reasoning benchmarks

1.2.1 ARC (AI2 Reasoning Challenge)

A set of questions and text passages collected for evaluating question-answering tasks that ask questions about science, with provided answer choices.

1.2.2 HellaSwag

A dataset used to evaluate reasoning abilities, with answer choices provided within the question set, containing over 10,000 pieces of data.

1.3 Coding Benchmarks

1.3.1 HumanEval

This dataset evaluates programming skills across more than 164 tasks, designed to assess model coding proficiency.

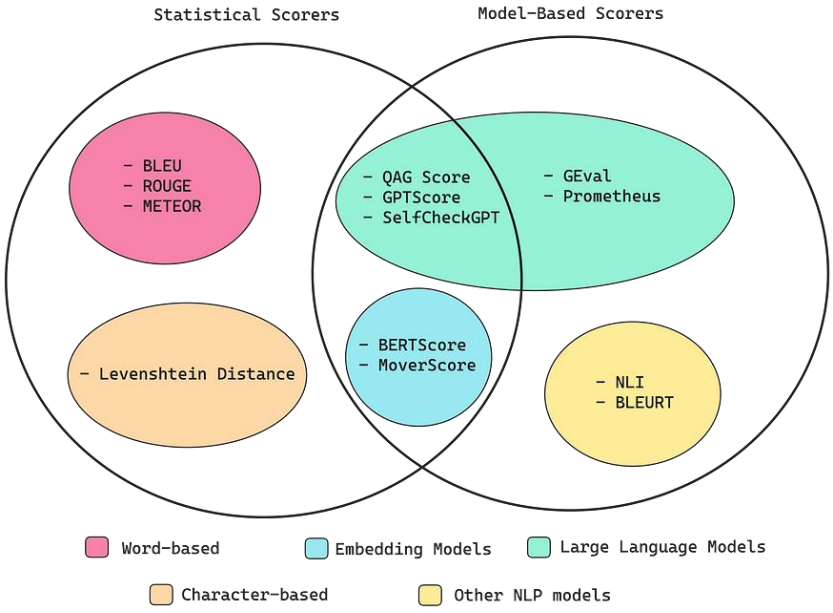
1.4 Limitation of Benchmarks

While evaluations are crucial for assessing the capabilities of LLMs (Language Models), they have limitations to consider:

1.Domain specificity: Evaluations often cannot fully adapt to specific domains or contexts where LLMs are utilized effectively, such as legal analysis or medical interpretation. This creates gaps in assessing the performance of LLMs in various specialized applications.

2. Age of standards: When new evaluation standards are introduced, LLMs often cannot perform as well as human benchmarks. Regular updates and new knowledge are necessary within 1-3 years.

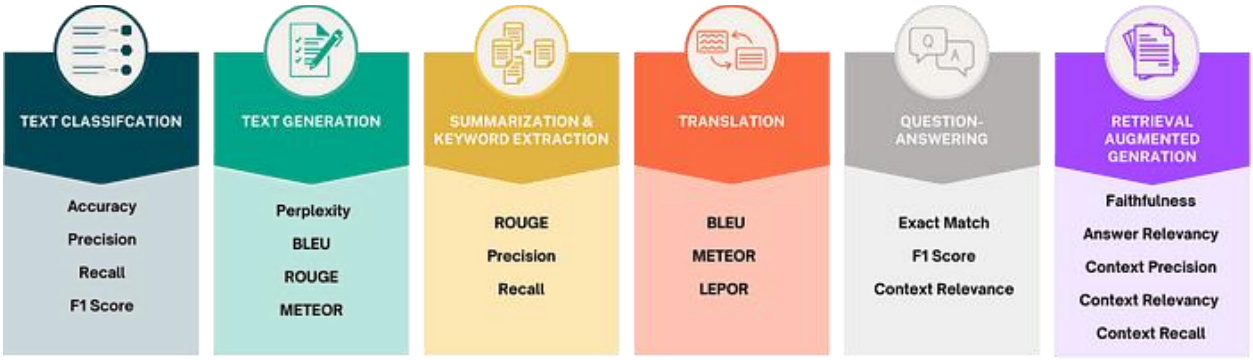
2. LLM Evaluation Metrics



Numerous established methods are available for calculating metric scores some utilize neural networks, including embedding models and LLMs, while others are based entirely on statistical analysis.

Task-dependent metrics are closely tied to the specific objectives of an LLM application. For example, text classification might prioritize accuracy and F1 scores, while text generation could focus on

perplexity, BLEU, or ROUGE scores depending on specific goals like readability or semantic accuracy. These metrics are chosen based on how closely they align with the desired outcome of the LLM's application.



2.1 Statistical Score

2.1.1 Word Error rate (WER)

WER is a family of WER-based metrics that measure the edit distance $d(c, r)$, i.e., the number of insertions, deletions, substitutions and possibly, transpositions required to transform the candidate into the reference string.

Word error rate can then be computed as:

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

where

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of correct words,
- N is the number of words in the reference (N=S+D+C)

2.1.2 Exact match

It measures the accuracy of candidate text by matching the generated text with the reference text. Any deviation from reference text will be counted as incorrect. This is only suitable for extractive and short-form answers where minimal or no deviation from the reference text is expected.

2.1.3 Perplexity

Perplexity (PPL) is one of the most common metrics for evaluating language models. Before diving in, we should note that the metric applies specifically to classical language models (sometimes called autoregressive or causal language models) and is not well-defined for masked language models like BERT

Perplexity is defined as the exponentiated average negative log-likelihood of a sequence. If we have a tokenized sequence $X = (x_0, x_1, \ldots, x_t)$, then the perplexity of X is,

$$\text{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}$$

where $\log p_\theta(x_i|x_{<i})$ is the log-likelihood of the ith token conditioned on the preceding tokens $x_{<i}$ according to our model. Intuitively, it can be thought of as an evaluation of the model's ability to predict uniformly among the set of specified tokens in a corpus. Importantly, this means that the tokenization procedure has a direct impact on a model's perplexity which should always be taken into consideration when comparing different models.

2.1.4 BLEU

The BLEU (BiLingual Evaluation Understudy) score is a widely used metric for evaluating the quality of machine-translated text (Candidate) against reference translations (Reference). Developed by IBM researchers, BLEU assesses translation accuracy by measuring the overlap of n-grams between the machine-generated text and a set of high-quality reference translations. It primarily focuses on precision. BLEU is renowned for its simplicity and effectiveness, making it a standard benchmark in the field of machine translation. However, it primarily evaluates surface-level lexical similarities, often overlooking deeper semantic and contextual nuances of language.

Example

BLEU Score:
Candidate Translation: The quick brown fox jumps over the lazy dog.
Unigrams=["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]= 9
Reference Translation: The quick brown dog jumps over the lazy fox.
Unigrams=["The", "quick", "brown", "dog", "jumps", "over", "the", "lazy", "fox"]=9
Unigram Precision
matching unigrams = 9 (all)

$$\textbf{Unigram Precision} = \frac{\text{(Overlapping 1-grams)}}{\text{(Total 1-grams in Candidate Translation)}} = \frac{9}{9} = 1$$

matching bigrams: ["The quick", "quick brown", "jumps over", "over the", "the lazy"]=5

$$\textbf{Bigram Precision} = \frac{\text{(Overlapping 2-grams)}}{\text{(Total 2-grams in Candidate Translation)}} = \frac{5}{8}$$

matching trigrams: ["The quick brown", "jumps over the", "over the lazy"] = 3

$$\textbf{Trigram Precision} = \frac{\text{(Overlapping 3-grams)}}{\text{(Total 3-grams in Candidate Translation)}} = \frac{3}{7}$$

matching 4-grams: ["jumps over the lazy"]

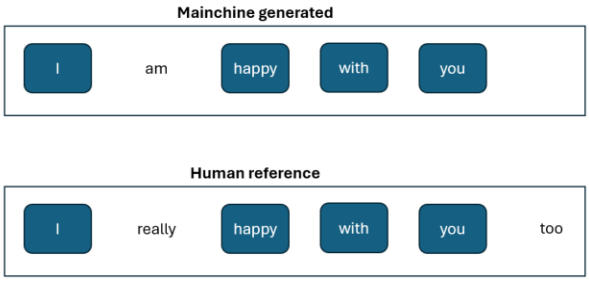
$$\textbf{4-gram Precision} = \frac{\text{(Overlapping 4-grams)}}{\text{(Total 3-grams in Candidate Translation)}} = \frac{1}{6}$$

PB = 1 since length is same
BLEU Score=1* exp((1/4) * (log(1) + log(5/8) + log(3/7) + log(1/6))) = 0.46

With Python Library:

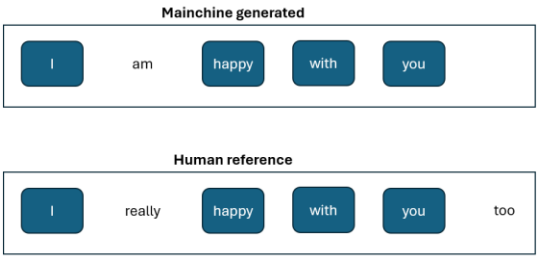
```
from nltk.translate.bleu_score import sentence_bleu

reference = [['The', 'quick', 'brown', 'fox', 'jumps',
'over', 'the', 'lazy', 'dog']]

candidate = ['The', 'quick', 'brown', 'dog', 'jumps',
'over', 'the', 'lazy', 'fox']


score = sentence_bleu(reference, candidate)


print(score)


0.46
```

Bilingual Evaluation Understudy: BLEU [Translation]



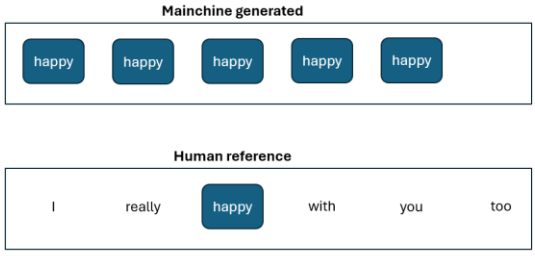
$$\text{Unigram Precision} = \frac{\text{No. of word matches}}{\text{No. of words generated}} = \frac{4}{5}$$

Bilingual Evaluation Understudy: BLEU [Translation]

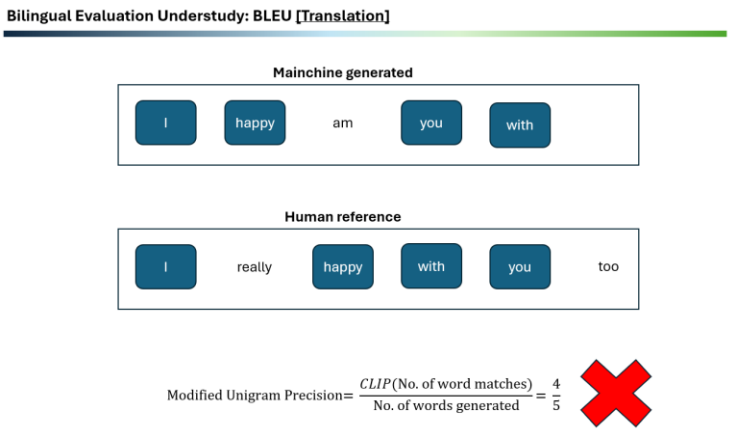


$$\text{Unigram Precision} = \frac{\text{No. of word matches}}{\text{No. of words generated}} = \frac{4}{5}$$

Bilingual Evaluation Understudy: BLEU [Translation]



$$\text{Modified Unigram Precision} = \frac{CLIP(\text{No. of word matches})}{\text{No. of words generated}} = \frac{1}{5}$$



Based on the example above, credited to P'ming, the strengths of this model include ease of use and speed, making it applicable not only to machine translation but also to other types of tasks. However, it has drawbacks in terms of deep semantic understanding, syntactic structure, and tokenization across diverse languages.

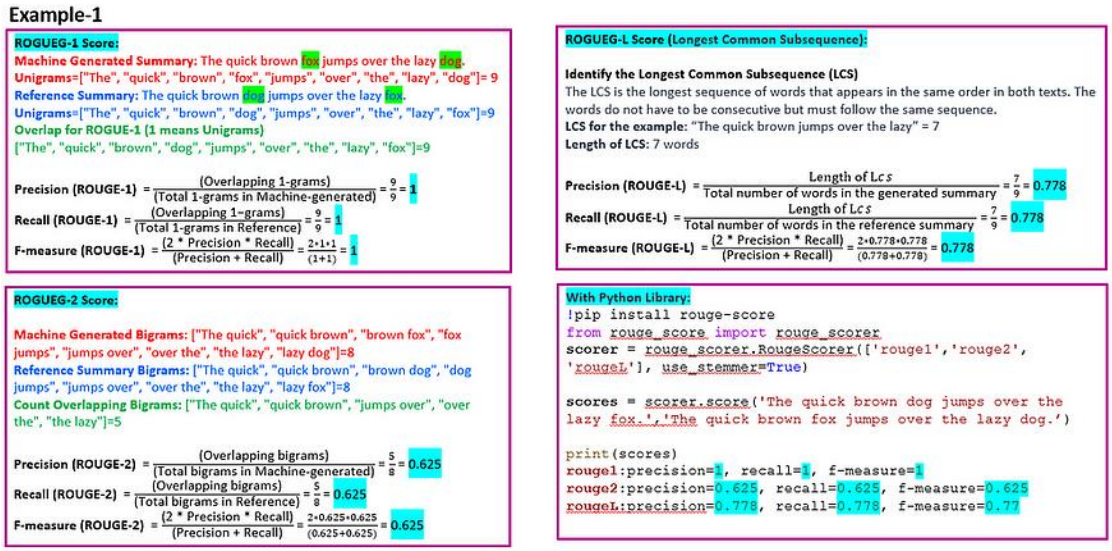
2.1.5 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used for evaluating automatic summarization and machine translation. It compares an automatically produced summary or translation against a set of reference summaries (usually human-written). ROUGE measures the quality of the summary by counting the number of overlapping units such as n-grams, word sequences, and word pairs between the model-generated text and the reference texts. The most common variants of ROUGE are

- ROUGE-N: Focuses on n-grams (N-word phrases). ROUGE-1 and ROUGE-2 (unigrams and bigrams, respectively) are most common.
- ROUGE-L: Based on the Longest Common Subsequence (LCS), which takes into account sentence level structure similarity naturally and identifies the longest co-occurring in-sequence n-grams automatically.

ROUGE typically reports three metrics.

- Precision: The proportion of the n-grams in the model-generated summary that are also found in the reference summary.
- Recall: The proportion of the n-grams in the reference summary that are also found in the model-generated summary.
- F-Score (F1 Score): The harmonic mean of precision and recall, balancing the two.



Although the ROUGE matrix is easy and fast to use, it still has limitations in understanding context and requires references for citation.

2.1.6 METEOR

METEOR (Metric for Evaluation of Translation with Explicit Ordering) is an advanced metric for evaluating machine translation that was developed to address some limitations of the BLEU score. Unlike BLEU, METEOR not only considers exact word matches but also incorporates stemming and synonyms to evaluate translations, thereby capturing a broader range of linguistic similarities. It uniquely balances precision and recall in its assessment and introduces a penalty for word order differences to evaluate the fluency of translations. METEOR is known for its higher correlation with human judgment, especially at the sentence level, making it a nuanced and comprehensive metric for translation quality evaluation. However, its sophistication also means it is more computationally intensive than simpler metrics like BLEU.

Calculate METEOR

- Count the number of unigrams in the candidate that exactly match the unigrams in the reference.
- Precision (P): The proportion of unigrams in the candidate translation that appear in the reference translation.
- Recall (R): The proportion of unigrams in the reference translation that appear in the candidate translation.
- Calculate the Harmonic Mean of Precision and Recall, The F-mean is calculated as: F-mean=10*P*R/(R+9*P). This places more weight on recall than precision.

- Penalty for Word Order, A penalty is applied for differences in word order. The penalty is calculated as: Penalty=0.5•(# of chunks/# of matches)**3; where a “chunk” is a set of adjacent words in the candidate that are in the same order as in the reference.
- The final score is computed as: Score=(1−Penalty)*F-mean

Example

METEOR Score:

Candidate Translation: The quick brown fox jumps over the lazy dog.
Unigrams=["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]= 9
Reference Translation: The quick brown dog jumps over the lazy fox.
Unigrams=["The", "quick", "brown", "dog", "jumps", "over", "the", "lazy", "fox"]=9
Total matches (overlap) = 9 words matches

Unigram Precision (P) $= \frac{\text{(Overlapping 1-grams)}}{\text{(Total 1-grams in Candidate Translation)}} = \frac{9}{9} = 1$

Unigram Recall (R) $= \frac{\text{(Overlapping 1-grams)}}{\text{(Total 1-grams in Reference Translation)}} = \frac{9}{9} = 1$

Harmonic (F) Mean $= \frac{10PR}{R+9P} = \frac{10*1*1}{1+9*1} = 1$

There are three chunks: (fox, dog) and (dog, fox) are the same chunk

Penalty $= 0.5*(\frac{\text{\# of chunks}}{\text{\# of matches}}) ** 3 = 0.5*(\frac{3}{7}) ** 3 =$ **0.03935**

Score = (1-Penalty)*Fmean =(1-0.03935)*1 = 0.961

With Python Library:

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
from nltk.translate.meteor_score import meteor_score
from nltk import word_tokenize

prediction = " The quick brown fox jumps over the lazy dog"
reference = " The quick brown dog jumps over the lazy fox"

results = meteor_score([word_tokenize(reference)], word_tokenize(prediction),
alpha = 0.9, beta = 3, gamma = 0.5)

Print(results)
0.956
```

Even though it's a metric developed from the BELU score, ROUGE still has limitations in evaluating languages other than English, including the requirement for consistent tokenization.

2.1.7 CiDER

CIDEr is a metric designed to evaluate the quality of image captions in image captioning tasks. It considers similarities at the word usage level, the relational structure of words, and the importance of key and contextually relevant words. CIDEr computes by comparing candidate captions with multiple reference captions. The calculation involves using TF-IDF to emphasize important and non-repetitive words in context. Subsequently, these TF-IDF values are used in cosine similarity to measure the similarity between the generated caption and reference captions. This process yields average CIDEr scores across various n-grams.

The advantage is its ability to match words more effectively as the number of reference captions increases. However, it necessitates an expanding list of reference captions.

https://www.youtube.com/watch?v=3nZF99Z4CIc

2.2 Model-Based Scorers

2.2.1 G-Eval

G-Eval is a recently developed framework using GPT-4 with Better Human Alignment” that uses LLMs to evaluate LLM outputs (aka. LLM-Evals).

G-Eval first generates a series of evaluation steps using chain of thoughts (CoTs) before using the generated steps to determine the final score via a form-filling paradigm (this is just a fancy way of saying G-Eval requires several pieces of information to work). For example, evaluating LLM output coherence using G-Eval involves constructing a prompt that contains the criteria and text to be evaluated to generate evaluation steps

2.2.2 Prometheus

Prometheus is a fully open-source LLM that is comparable to GPT-4's evaluation capabilities when the appropriate reference materials (reference answer, score rubric) are provided. It also uses case agnostic, similar to G-Eval. Prometheus is a language model using Llama-2-Chat as a base model and fine-tuned on 100K feedback (generated by GPT-4) within the Feedback Collection.

2.3 Combining Statistical and Model-Based Scorers

2.3.1 GPTScore

Unlike G-Eval which directly performs the evaluation task with a form-filling paradigm,

https://wandb.ai/vincenttu/blog_posts/reports/Evaluating-Generative-Models-with-GPTScore--VmlldzozNTI2NjQy

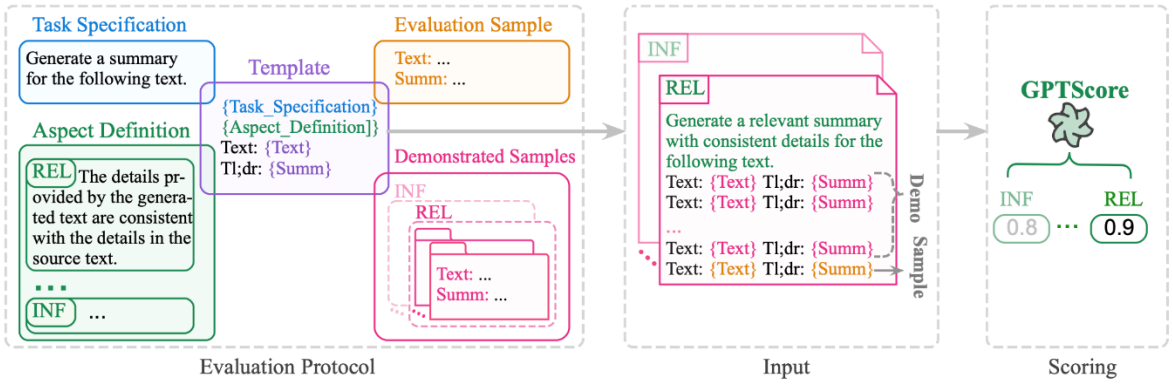
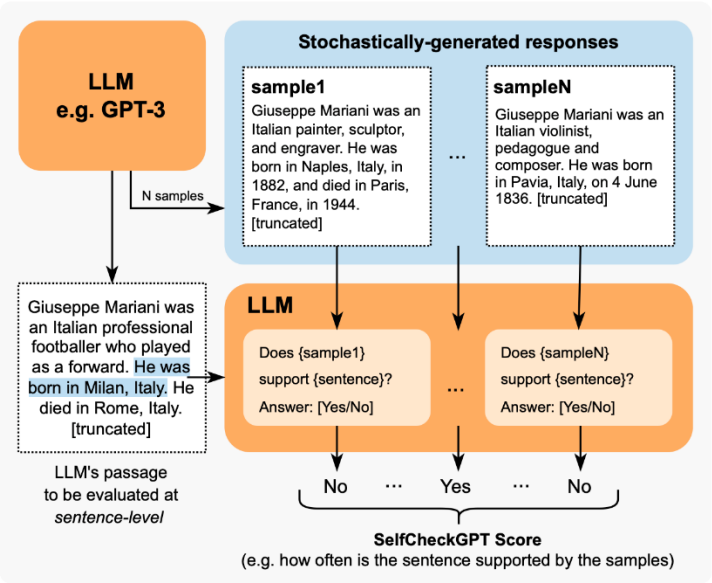


Figure 2. The framework of GPTSCORE. We include two evaluation aspects *relevance (REL)* and *informative (INF)* in this figure and use the evaluation of *relevance (REL)* of the text summarization task to exemplify our framework.

2.3.2 SelfCheckGPT

SelfCheckGPT is an odd one. It is a simple sampling-based approach that is used to fact-check LLM outputs. It assumes that hallucinated outputs are not reproducible, whereas if an LLM has knowledge of a given concept, sampled responses are likely to be similar and contain consistent facts. SelfCheckGPT is an

interesting approach because it makes detecting hallucination a reference-less process, which is extremely useful in a production setting.



2.3.3 QAG Score

QAG (Question Answer Generation) Score is a scorer that leverages LLMs' high reasoning capabilities to reliably evaluate LLM outputs. It uses answers (usually either a 'yes' or 'no') to close-ended questions (which can be generated or preset) to compute a final metric score. It is reliable because it does NOT use LLMs to directly generate scores. For example, if you want to compute a score for faithfulness (which measures whether an LLM output was hallucinated or not), you would:

1. Use an LLM to extract all claims made in an LLM output.
2. For each claim, ask the ground truth whether it agrees ('yes') or not ('no') with the claim made.

Example LLM output:

Martin Luther King Jr., the renowned civil rights leader, was assassinated on April 4, 1968, at the Lorraine Motel in Memphis, Tennessee. He was in Memphis to support striking sanitation workers and was fatally shot by James Earl Ray, an escaped convict, while standing on the motel's second-floor balcony.

A claim would be:

Martin Luther King Jr. assassinated on the April 4, 1968

And a corresponding close-ended question would be:

Was Martin Luther King Jr. assassinated on the April 4, 1968?

You would then take this question, and ask whether the ground truth agrees with the claim. In the end, you will have a number of 'yes' and 'no' answers, which you can use to compute a score via some mathematical formula of your choice.

2.4 RAG Metrics

2.3.1 Faithfulness

Faithfulness is a RAG metric that evaluates whether the LLM/generator in your RAG pipeline is generating LLM outputs that factually aligns with the information presented in the retrieval context.

1. Use LLMs to extract all claims made in the output.
2. For each claim, check whether the it agrees or contradicts with each individual node in the retrieval context. In this case, the close-ended question in QAG will be something like: "Does the given claim agree with the reference text", where the "reference text" will be each individual retrieved node. (Note that you need to confine the answer to either a 'yes', 'no', or 'idk'. The 'idk' state represents the edge case where the retrieval context does not contain relevant information to give a yes/no answer.)
3. Add up the total number of truthful claims ('yes' and 'idk'), and divide it by the total number of claims made

2.3.2 Answer Relevancy

Answer relevancy is a RAG metric that assesses whether your RAG generator outputs concise answers, and can be calculated by determining the proportion of sentences in an LLM output that a relevant to the input (ie. divide the number relevant sentences by the total number of sentences).

2.3.3 Contextual Precision

Contextual Precision is a RAG metric that assesses the quality of your RAG pipeline's retriever. When we're talking about contextual metrics, we're mainly concerned about the relevancy of the retrieval context. A high contextual precision score means nodes that are relevant in the retrieval contextual are ranked higher than irrelevant ones. This is important because LLMs gives more weighting to information in nodes that appear earlier in the retrieval context, which affects the quality of the final output.

2.3.4 Contextual Recall

Contextual Precision is an additional metric for evaluating a Retriever-Augmented Generator (RAG). It is calculated by determining the proportion of sentences in the expected output or ground truth that can be attributed to nodes in the retrieval context. A higher score represents a greater alignment between the retrieved information and the expected output, indicating that the retriever is effectively sourcing relevant and accurate content to aid the generator in producing contextually appropriate responses.

2.3.5 Contextual Relevancy

Probably the simplest metric to understand, contextual relevancy is simply the proportion of sentences in the retrieval context that are relevant to a given input.

2.5 Model-Based Scorers

2.3.1 BLEURT (Bilingual Evaluation Understudy with Representations from Transformers)

BLEURT (Bilingual Evaluation Understudy with Representations from Transformers) is a novel, machine learning-based automatic metric that can capture non-trivial semantic similarities between sentences. It is trained on a public collection of ratings (the WMT Metrics Shared Task dataset) as well as additional ratings provided by the user.

Input: Bud Powell était un pianiste de légende. **Reference:** Bud Powell was a legendary pianist.	**BLEURT**
Candidate 1: Bud Powell was a legendary pianist.	*1.01*
Candidate 2: Bud Powell was a historic piano player.	*0.71*
Candidate 3: Bud Powell was a New Yorker.	*-1.49*

The limitation of the matrix relies heavily on the model's capabilities and its applicability, which may only be effective within the domain for which the model was trained. This might not be suitable for current use cases and lacks recent updates.

2.6 Fine-tuning metrics

“fine-tuning metrics”, mean is metrics that assess the LLM itself, rather than the entire system. Putting aside cost and performance benefits, LLMs are often fine-tuned to either:

1. Incorporate additional contextual knowledge.
2. Adjust its behavior.

2.6.1 Hallucination

Some of you might recognize this being the same as the faithfulness metric. Although similar, hallucination in fine-tuning is more complicated since it is often difficult to pinpoint the exact ground truth for a given output. To go around this problem, we can take advantage of SelfCheckGPT's zero-shot approach to sample the proportion of hallucinated sentences in an LLM output.

2.6.2 Toxicity

The toxicity metric evaluates the extent to which a text contains offensive, harmful, or inappropriate language. Off-the-shelf pre-trained models like Detoxify, which utilize the BERT scorer, can be employed to score toxicity.

2.6.3 Bias

The toxicity metric evaluates the extent to which a text contains offensive, harmful, or inappropriate language. Off-the-shelf pre-trained models like Detoxify, which utilize the BERT scorer, can be employed to score toxicity.

3. Evaluation tool or framework

3.1 DeepEvals

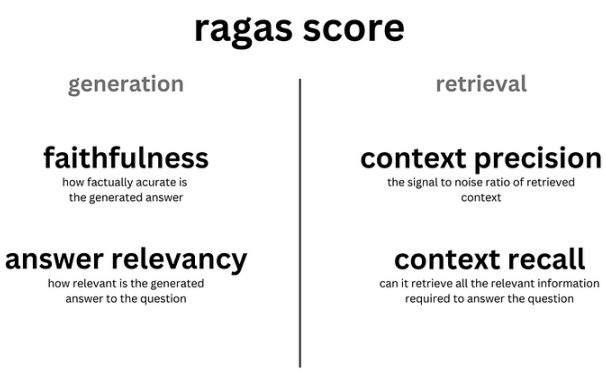
DeepEvals is a framework specifically focused on evaluation related to NLP tasks, gathering various important matrices suitable for current NLP developments. It also supports additional data synthesis and unit testing.

The framework's strengths include matrices that support current NLP tasks such as question answering, making it suitable for evaluating LLMs.

3.2 Langsmith

Langsmith is a framework designed primarily for monitoring the outcomes of LLM processing operations. However, Langsmith may not provide extensive evaluation capabilities and might need integration with other libraries like LangChain or DeepEvals for evaluation purposes. Its advantage lies in its ability to trace each transaction.

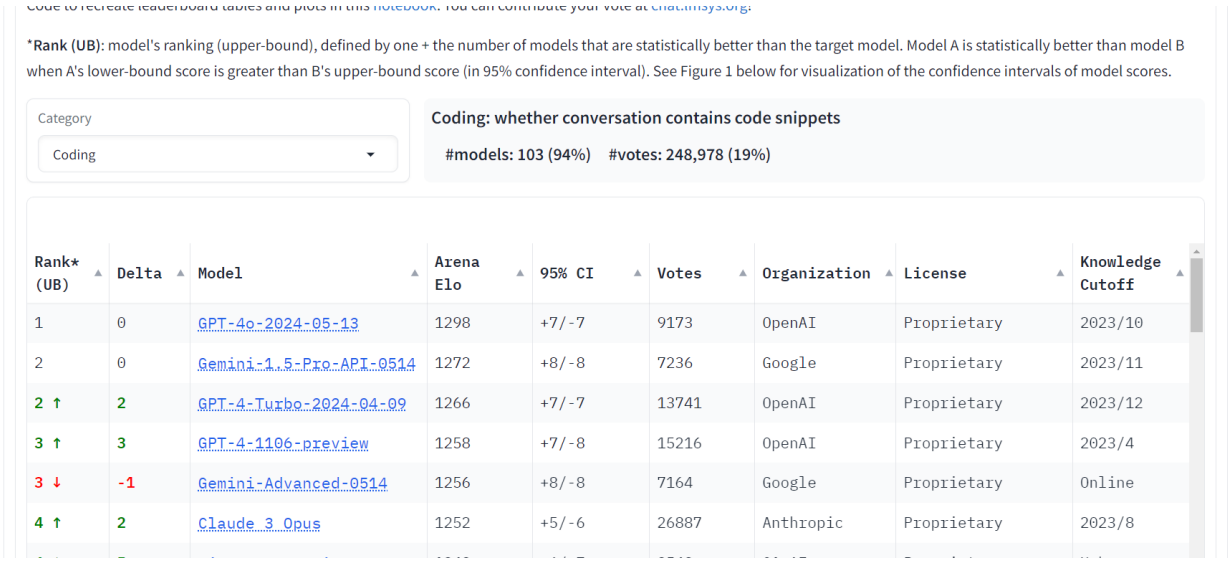
3.3 RAGAS



This is a framework that integrates previously mentioned metrics such as faithfulness, context precision, answer relevancy, and context recall into an end-to-end evaluation format. This integration aims to facilitate straightforward application with RAG applications.

4. How to select LLM evaluation

One crucial factor when considering the use of LLM models for evaluation is their ability to comprehend language effectively and exhibit human-like reasoning or conceptual understanding. This ensures that the model's outputs closely align with human evaluations, thus aiding in decision-making and model selection for assessment tasks. Therefore, it's essential to refer to leaderboards that compare various models on websites like https://chat.lmsys.org/?leaderboard. Each model may specialize in different tasks, so evaluating their performance across these tasks is vital.



Code to recreate leaderboard tables and plots in this notebook. You can contribute your vote at chat.lmsys.org!

*Rank (UB): model's ranking (upper-bound), defined by one + the number of models that are statistically better than the target model. Model A is statistically better than model B when A's lower-bound score is greater than B's upper-bound score (in 95% confidence interval). See Figure 1 below for visualization of the confidence intervals of model scores.

Category: Coding

Coding: whether conversation contains code snippets

#models: 103 (94%) #votes: 248,978 (19%)

Rank* (UB)	Delta	Model	Arena Elo	95% CI	Votes	Organization	License	Knowledge Cutoff
1	0	GPT-4o-2024-05-13	1298	+7/-7	9173	OpenAI	Proprietary	2023/10
2	0	Gemini-1.5-Pro-API-0514	1272	+8/-8	7236	Google	Proprietary	2023/11
2 ↑	2	GPT-4-Turbo-2024-04-09	1266	+7/-7	13741	OpenAI	Proprietary	2023/12
3 ↑	3	GPT-4-1106-preview	1258	+7/-8	15216	OpenAI	Proprietary	2023/4
3 ↓	-1	Gemini-Advanced-0514	1256	+8/-8	7164	Google	Proprietary	Online
4 ↑	2	Claude 3 Opus	1252	+5/-6	26887	Anthropic	Proprietary	2023/8

Reference :

- https://medium.com/@vipra_singh/building-llm-applications-evaluation-part-8-fcfa2f22bd1c
- https://www.confident-ai.com/blog/llm-evaluation-metrics-everything-you-need-for-llm-evaluation
- https://klu.ai/glossary/llm-evaluation