LLM Metrics

- Evaluating a language model helps to understand its performance on various tasks and in various domains.

→ Performance metrics like precision, recall, perplexity etc provide a more quantitative summary of how well a model is performing on language related tasks.

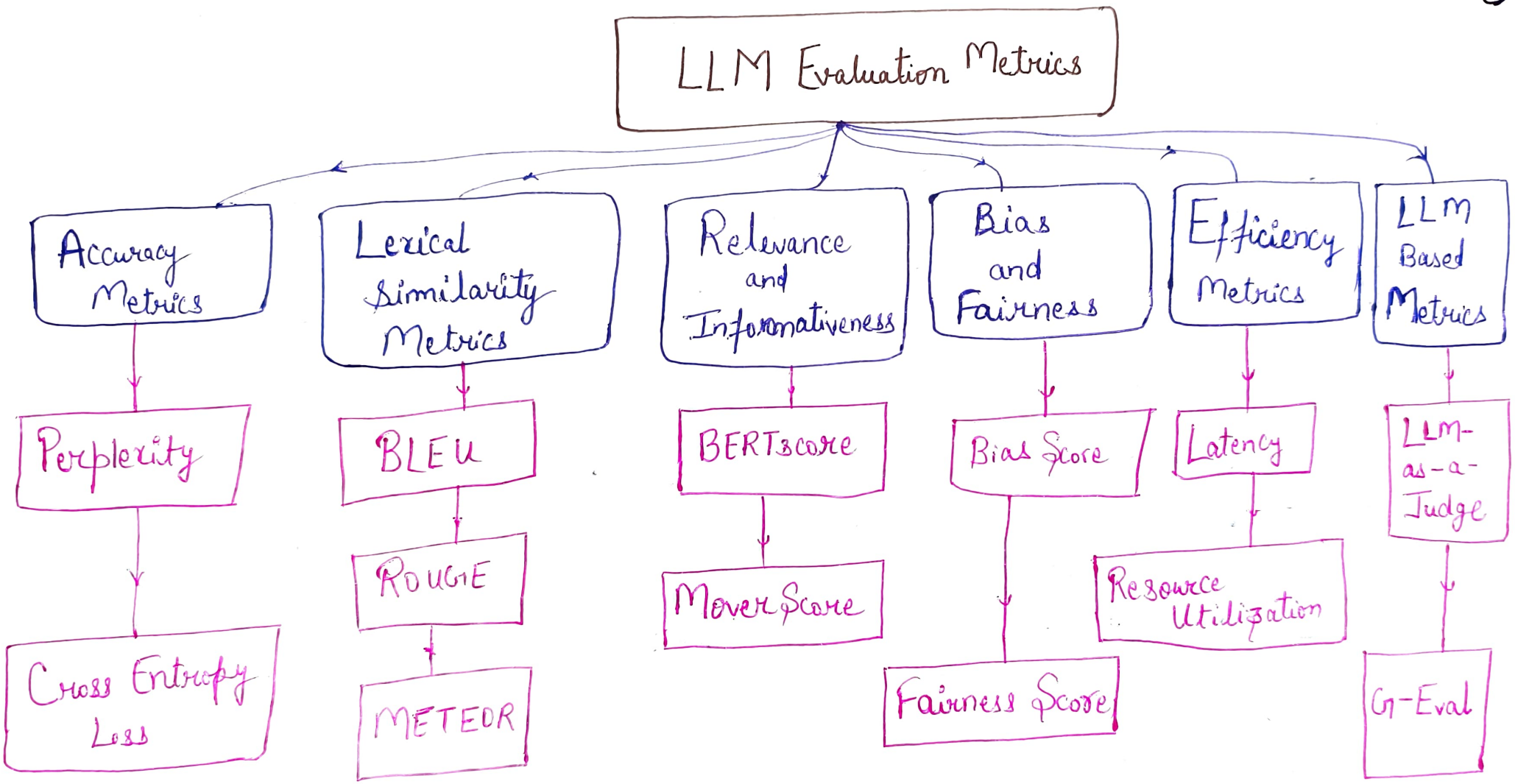
- There metrics help in determining if selected model meets specified performance or falls short in handling the intricacies of human language reliably.

- **Fairness and Bias Detection :-**

→ Evaluation of LLMs for fairness and bias helps in developing ethical systems.

. This aspect of evaluation is essential for building inclusive technologies that respect diversity and avoid harm.

- Evaluation metrics provide a feedback loop that guides the iterative process of model development and refinement.


LLM Evaluation Metrics
Accuracy Metrics
Lexical Similarity Metrics
Relevance and Informativeness
Bias and Fairness
Efficiency Metrics
LLM Based Metrics
Perplexity
Cross Entropy Loss
BLEU
ROUGE
METEOR
BERTscore
MoverScore
Bias Score
Fairness Score
Latency
Resource Utilization
LLM-as-a-Judge
G-Eval

Accuracy Metrics :-

(a) **Cross-entropy loss** : It measure difference b/w true probability distribution of target words and predicted probability distribution produced by the model.

Idea is to quantify the likelihood of model's predictions matching the Actual target.

↓↓ cross entropy = closer match { Actual / Predicted } sequence of words

For a given sequence of words $w_1, w_2 \ldots w_N$,

where $P\left(\frac{w_i}{w_{<i}}\right)$ → model's predicted probability of words w_i given previous words $w_{<i}$.

Cross-Entropy H :—

$$H = -\frac{1}{N}\sum_{i=1}^{N} \log P\left(\frac{w_i}{w_{<i}}\right)$$

→ previous words

eg:- Sequence of words : "The cat sat on the mat".

Assume our language model assign following predicted probabilities for each word given previous words :—

- P("The") = 0.2
- P("cat" | "The") = 0.1
- P("sat" | "The cat") = 0.15
- P("on" | "The cat sat") = 0.3
- P("the" | "The cat sat on") = 0.25
- P("mat" | "The cat sat on the") = 0.05

Step 1:- Let's calculate Cross-entropy as follows :— (Compute the Log probabilities) :—

- log P("The") = log(0.2) = -1.6094
- log P("cat" | "The") = log(0.1) = -2.3026
- log P("sat" | "The cat") = log(0.15) = -1.8971
- log P("on" | "The cat sat") = log(0.3) = -1.2039
- log P("the" | "The cat sat on") = log(0.25) = -1.3
- log P("mat" | "The cat sat on the") = log(0.05) = -2.9957

Step 2: Sum the log probabilities :-

$$\sum \log P(w_i) = -1.6094 + -2.3026 + -1.8971 + -1.2039 + -1.3863 + -2.9957$$

$$= -11.3950$$

Step 3: Compute the average negative log probability :-

$$-\frac{\sum \log P(w_i)}{6} = \frac{-11.3950}{6} = 1.8992$$

- Lower Cross-entropy loss indicate :-

predicted probability distribution is closer to true distribution of the target words.

↓

better model performance.

(b) **Perplexity** :- It quantifies how well a probabilistic model predicts a sample. (a piece of text in this case).

- It is measure of how uncertain a model is in predicting the next word in the sequence.

> It is measure of amount of randomness in the model

eg: If perplexity for a model = 4

↓

Then model had a 1-in-4 chance of guessing (on avg) the next word in the text.

> lower perplexity score ⇒ better model performance
>
> as model will have to choose from a small set of words
>
> ↓
>
> to predict next word

- Perplexity depends on cross-entropy loss and

→ represents average branching factor of a predictive model, indicating how many possible next words it considers.

Although model assigns a probability score to all possible available words in corpus.

↓

perplexity only pays attention to top N words that are being considered as the potential next word.

- Perplexity depends on same data that was originally used to train the model

↓

as it measure extent to which model learned the data distribution.

↓

Thus, it can only be used to compare models trained on same data.

eg:- If a language model assigns a probability $P(w_1, w_2 \ldots w_N)$ to a sequence of words $w_1, w_2 \ldots w_N$.

Perplexity PP :—

$$PP(W) = P(w_1, w_2 \ldots \ldots w_N)^{-\frac{1}{N}}$$

Alternatively, it can be expressed as avg. negative log-likelihood of words in sequence.

$$PP(w) = \exp\left(-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i | w_1, w_2 \ldots w_{i-1})\right)$$

eg:- from previous cross entropy example :-

$$PP(w) = \exp(1.89992) = 6.688$$

- ↑↑ Perplexity value is ideal for use cases that require higher creativity for ex. generating poetry, fiction or marketing copy.

- ↓↓ perplexity ⇒ for task specific applications like machine translation, summarization etc.

Lexical Similarity

(a) BLEU (Bilingual Evaluation Understudy)

- precision based metric
- Primarily used for evaluating quality of text generated by the LLMs.

→ It was primarily designed for language translation tasks but it can be extended to other tasks like text summerization, text generation etc.

- BLEU compares the n-gram of the generated text to one or more reference (ground-truth) texts

Note: N-grams are continuous sequence of words or tokens in a document.

1-gram = Single token

2-grams = Combination of 2 words

⋮

n-grams = Collection of n numbers of tokens.

→ It starts with calculating precision for each matching n-gram in reference (human-crafted) text and model-generated text.

→ Then it applies brevity penality to adjust the score to penalize shorter translations / text that might achieve high precision by being too short.

- BLEU score is calculated for different n-grams length (normally 4) and then combined into a single score by calculating the geometric mean.

eg:- Candidate (generated) translation :- "The quick brown fox jumps over the lazy dog"

- Reference translation :- "A fast brown fox leaps over the lazy dog".

Step 1 :- Calculate precision (P_n) for each n-gram length :-

$$P_n = \frac{\text{no. of n-gram in candidate that match any of the reference translations}}{\text{Total no. of n-grams in the candidate}}$$

1-gram precision : Count the matching unigrams.

- Matching unigrams : "brown", "fox", "over", "the", "lazy", "dog" (6 matches)

Total unigrams in candidate : 9

$$P_1 = \frac{6}{9} = 0.6667$$

2-gram precision : Count matching bigrams.

- Matching bigrams : "brown fox", "over the", "the lazy", "lazy dog" (4 matches)

Total bigrams in Candidate : 8

The quick brown fox jumps over the lazy dog
1 2 3 4 5 6 7 8

$$P_2 = \frac{4}{8} = 0.5$$

3-gram precision : Count matching trigrams

- Matching trigrams : "over the lazy", "the lazy dog" (2 matches)

Total trigrams in candidate : 7

$$P_3 = \frac{2}{7} = 0.2857$$

4 gram precision :- Count the matching 4-grams.

Matching 4-grams : "the lazy dog" (1 match)

Total 4-grams in candidate : 6

$$P_4 = \frac{1}{6} = 0.1667$$

Step 2 : Combine the precision scores using the geometric mean :

$$P_4 = \left(P_1 \times P_2 \times P_3 \times P_4\right)^{\frac{1}{4}}$$

$$= \left(0.6667 \times 0.5 \times 0.2857 \times 0.1667\right)^{\frac{1}{4}} = 0.349$$

Step 3 : Apply a brevity penalty (BP) :-

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1 - \frac{r}{c}\right)} & \text{if } c \leq r \end{cases}$$

In this case, both candidate and reference translations have 9 words, so $BP = 1$

Step 4 : Calculate BLEU score :-

$$BLEU = BP * P_4$$

$$= 1 * 0.349 = 0.349$$

Blue score rang : 0 to 1.

Higher blue score ↑↑ = better quality

↘ closer alignment with the reference texts.

Drawbacks of BLEU score :-

(i) Bleu relies on exact n-gram matching and doesn't take into account the context and meaning of the text leading to potential inaccuracy in the evaluation.

(ii) Synonyms and other similar words are not recognized by BLEU which leads to generating lower scores even if sentences are similar.

(iii) BLEU often fails to capture the longer dependencies and structure as it evaluates the n-gramms upto a certain length (typically 4).

- doesn't consider meaning
- relies on exact n-gram matching
- Synonyms, similar words not considered
- fails to capture longer dependencies.

ROUGE (Recall-oriented Understudy for Gisting Evaluation):-

- Rouge is most popular evaluation method that is majorly used for text summerization tasks and evaluates quality of LLM generated summaries by comparing them to reference summeries.

- Unlike precision focused metrics like BLEU,
  ↓
  ROUGE emphasizes recall
  ↓
  measuring how much of reference summary is captured by generated summary.

→ It calculates the percentage (0-1) of reference's n-grams that are included in the LLM output.

ROUGE has multiple variants!—

- ROUGE-N → matches b/w different ngrams.
- ROUGE-L → Captures sequential similarity
- ROUGE-W : weighted version of ROUGE-L
- ROUGE-S : Calculates skip-bigram co-occurance statistics

ROUGE has multiple variants including :-

(a) ROUGE-N :- It measures match b/w different n-grams in reference and candidate summaries.

→ ROUGE-1 compares a single token at a time,

→ ROUGE-2 compares a combination of 2 tokens in the reference and candidate summeries and so forth.

(b) ROUGE-L :- It measures the longest common subsequences (LCS) b/w the generated and reference texts, capturing the sequence similarity.

(c) ROUGE-W :- A weighted version of ROUGE-L, giving more weight to consecutive matches.

(d) ROUGE-S :- It calculates skip-bigram co-occurance statics. Word pairings that maintain their sentence structure but allow arbitrary breaks are known as skip-bigrams.

eg :- To calculate ROUGE-N metrics summary :

· Candidate Summary :- "The quick brown fox jumps over the lazy dog"

Reference Summary : "The fast brown fox leaps over the lazy dog"

ROUGE-N = $\frac{\text{no. of matching n-grams}}{\text{Total no. of n-grams in reference summary.}}$

Step 1 :- Calculate ROUGE-1 (unigram overlap)

- Unigram in the candidate :

{"The", "quick", "brown", "fox", "jumps", "over", "the", "lazy" "dog"}

Unigrams in reference :-

{"The", "fast", "brown", "fox", "leaps", "over", "the", "lazy", "dog"}

Matching Unigrams : {"The", "brown", "fox", "over", "the", "lazy", "dog"}

: 7 matches

Total unigram in reference : 9

ROUGE-1 : $\frac{7}{9} = 0.7778$

Step 2 :- Calculate ROUGE-2 (bigram overlap) :-

- Bigrams in candidate : {"The quick", "quick brown", "brown fox", "fox jumps", "jumps over", "over the", "the lazy", "lazy dog"}

Bigram in reference : {"The fast", "fast brown", "brown fox", "fox leaps" "leaps over", "over the", "the lazy", "lazy dog"}

Matching bigrams :- { "brown fox", "over the", "the lazy", "lazy dog" } 4 matches

Total bigrams in reference : 8

ROUGE-2 :- $\frac{4}{8} = 0.5$

ROUGE score always has a value between 0 to 1.

1 : exact match of candidate and reference summaries.

Drawback of ROUGE

(i) Since ROUGE is recall-oriented,

⇓

it can lead to higher score for longer summaries

⇓

that include more of reference text even if they contain irrelevent information.

(ii) Similar to BLEU,

ROUGE also focuses on surface-level matching of the texts and can miss the deeper semantics similarities.

(iii) ROUGE also does not take into account the synonyms and similar words leading to poor score for texts that have same meaning but different words.

METEOR (Metric for evaluation of Translation with Explicit Ordering) :-

- BLEU and ROUGE don't take into account synonyms of words.
  - if 2 sentences have different words but same meaning → BLEU, ROUGE provide low score.

METEOR considers synonyms, stemming, and paraphrases in addition to exact words matching.

- Meteor score is based on harmonic mean (F-mean) of unigram precision and recall, with recall weighted higher than precision.

→ The metric also incorporates a penalty for incorrect word order.

Step 1 :- Match unigrams :-

Consider exact matches, stemming matches, synonym matches and paraphrase matches.

Step 2: Calculate precision and recall :

$$\text{Precision} = \frac{\text{no of matches}}{\text{Total no of unigrams in Candidate}}$$

$$\text{Recall} = \frac{\text{no of matches}}{\text{Total no. of unigrams in reference}}$$

Step 3:- Calculate F-mean score :-

$$Fmean = \frac{(10 * Precision * Recall)}{(9 * Precision + Recall)}$$

Step 4:- Calculate the penalty for word order :-

$$Penalty = 0.5 \left(\frac{num\_Chunks}{num\_matches} \right)$$

Step-5 :- Combine F-mean score and penality :-

$$METEOR = Fmean(1 - penality)$$

→ It provide nuanced evaluation compared to BLEU etc.

0 : no match

1 : perfect match

eg :- reference text : "The cat is sitting on the mat."

generated text :- "A cat is sitting on a mat."

METEOR Score Calculation :-

(i) Exact match : "cat", "is", "sitting", "on", "mat"

(ii) Stemming | Synonyms :- 'A' and "The" are considered interchangeable, so there's a match.

(iii) Precision :- 6 out of 7 words in generated answer match reference.

(iv) Recall :- 6 out of 6 words in reference match generated answer.

$$\text{Precision} = \frac{6}{7} = 0.857 \quad , \quad \text{Recall} = \frac{6}{6} = 1.0$$

(v) F_1 Score :- harmonic mean of Precision and Recall :-

$$F_1 = 2\left(\frac{0.857 * 1.0}{0.857 + 1.0}\right) = 0.923$$

(vi) Fragmentation Penality :- Since word order is similar with minimal fragmentation, the penality is small.

Let's assume a penality = 0.05

(vii) Final METEOR :- It is calculated by substracting penalty from F_1 score.

$$\text{METEOR} = 0.923 - 0.05 = \frac{0.873}{1}$$

score indicates high level of similarity.

Relevance and Informativeness :-

(a) **BERTScore** :- It is popular metric for evaluating the quality of text generated by language models,

↓

leveraging contextual embeddings from pre-trained BERT language model.

(Bidirectional Encoder Representations from Transformers)

- While traditional metrics rely on token match

↓

Bertscore use Cosine-Similarity b/w embedding of generated and reference text.

- This allows BERTscore to capture semantic similarity and nuances that n-gram based metric might miss.

Steps to Calculate BERTscore :-

(a) Generate embeddings
(use a pre-trained BERT model to generate embedding for each token in candidate and reference texts)

(b) Calculate Cosine Similarity
↓
b/w each token in candidate and most similar token in the reference.

(c) Aggregate scores
Aggregate the similarity scores to get :-
Precision, recall and F1 score.

Example for BERTScorer

(i) Compute Embeddings

(ii) Match Tokens

(iii) Calculate Precision, Recall

$$\text{Precision} = \frac{1}{|T_g|} \sum_{t_g \in T_g} \max_{t_r \in T_r} \text{cosine-similarity}(t_g, t_r)$$

$$\text{Recall} = \frac{1}{|T_r|} \sum_{t_r \in T_r} \max_{t_g \in T_g} \text{cosine-similarity}(t_r, t_g)$$

t_g :- set of tokens in generated text

t_r : " " " reference text

(iv) Compute F1 score $= 2 * \frac{P * R}{P + R}$

eg:- generated Text : "The cat sat on the mat"

Reference Text : "A cat was sitting on the mat"

(i) Compute Token embeddings :- assume Bert embedding for each token are Vectors

(ii) Compute Cosine Similarity

(iii) match tokens and calculate Precision, Recall

"The" : generated text , "A" : reference text

"cat" → "cat"

sat → sitting

"on" : generated	"on" : reference text
"the" "	"the"
mat	mat

- **Precision Calculation**: Average cosine similarity for each token in generated text.

"The" vs "A". : similarity = 0.8

"Cat" vs "cat" " = 1.0

"sat" vs "sitting" = 0.6

"on" vs "on" = 1.0

"the" vs "the" = 1.0

"mat" vs "met" = 1.0

$$\text{Precision} = \frac{0.8 + 1.0 + 0.6 + 1.0 + 1.0 + 1.0}{6} = \approx 0.9$$

Recall :-

"A" vs "The" = 0.8

"at" " "cat" = 1.0

"sitting" " "sat" = 0.6

"on" " "on" = 1.0

"the" " "the" = 1.0

"mat" " "mat" = 1.0

$$\text{Recall} = (0.8 + 1.0 + 0.6 + 1.0 + 1.0 + 1.0)/6 = 0.9$$

F_1 Score Calculation:

$$F_1 = \frac{2 * 0.9 * 0.9}{0.9 + 0.9} = 0.9$$

(b) **MoverScore:** It measure quality of text generation by calculating distance required to transform one text into another. (reference) (1 candidate)

→ It combines strength of both embedding based and distance-based approaches.

→ Similar to BERTScore, it leverages word embeddings to capture semantic meaning, making it more robust to various variations in wording.

- It is based on the concept of Earth Mover's Distance (EMD) which calculates minimum cost to transform one distribution to another.

Here: Work needed to move the words

Words in Candidate translations —moved to match→ reference translation.

Steps to Calculate MoverScore:

(a) Generate embeddings
(b) Compute pairwise distances
(c) Solve optimal transport problem.

(i) Generate Embeddings :- use a pre-trained (like bert) to generate word embeddings for both candidate and reference texts.

(ii) Compute pairwise distance :- Calculate cosine distance b/w all pairs of word embeddings from candidate and reference texts.

(iii) Solve optimal transport problems :- Use EMD to find minimal cost of transforming candidate word distribution to reference word distribution.

Generated text : "The quick brown fox".

Reference text : "A fast brown fox".

(i) Compute Embeddings : Let's assume 2D vector, embedding for each word in 2D vector :-

"The" : [0.1, 0.2]

"quick" : [0.4, 0.3]

"brown" : [0.2, 0.5]

"fox" : [0.3, 0.4]

"A" : [0.0, 0.1]

"fast" : [0.5, 0.4]

"brown" : [0.2, 0.5]

"fox" : [0.3, 0.4]

(ii) Compute pairwise distances :-
eg :- "quick", "fast" or "The", "A"

(iii) Optimal Transport :- matching words while minimizing 'cost' Distance :-

"The to "A" = 0.1

"quick" to "fast" = 0.2

"brown" to "brow" = 0.0

"fox" to "fox" : 0.0

(iv) Score Calculation :-

$0.1 + 0.2 + 0.0 + 0.0 = 0.3$

Model-Based Evaluation :-

(a) **LLM-as-a Judge :-** while automatic evaluation metrics like ROUGE, BLUE etc are pretty good in evaluating the model performance, they still require a human involved in the loop.

- Ground truth must always be created by human experts and final result of metrics also needs to be verified.
- LLM-as-a Judge provides end-to-end solution to evaluating LLM models, especially the RAG pipelines.

It involves following stages to evaluate the model :-

(i) Generate a synthetic dataset for evaluation using a text corpous with the help of an LLM model.

(ii) Use Another LLM (called a critic agent) to filter out most relevant data points in data.

(iii) Ask LLM to compare model-generated results and ground truths

→ provide score b/w 1 to 5 based on relevancy, groundness.

This method involves employing an LLM to provide impartial, consistent, comprehensive evaluation of another model's o/p across various task, metrics.

(6) **G-Eval** :- It is another model-based evaluation metric.

→ widely used for text summerization tasks.

- This approach leverages capabilities of LLM to provide detailed, and context aware evaluations, making it a powerful tool for comprehensive model assessment.

Steps involved in G-Eval :-

(a) **Task generation** :- Develop a set of prompts or tasks that cover a wide range of use cases, scenario and difficulty levels.

These tasks should be representative of real-world applications of LLM.

(b) **Output Generation** :- Use LLM to generate responses to the tasks.

Ensure the prompts are varied to test model's versatility and adaptability.

(c) **Automated Scoring** :- Apply automated metrics such as BLEU, ROUGE, METEOR and BERTscore to generated outputs.

→ these metrics provide quantitative measures of o/p quality.

(d) **Human-Like Evaluation** :- Use a judging LLM or human evaluators to provide qualitative assessments.

This step provide more subjective criteria as coherence, fluency, and relevance to the context.

(e) Feedback generation :- Compile results from both automated and human-like-evaluations into a comprehensive report.

Report should highlight models strength, weaknesses and provide actionable feedback for improvement.

Bias and Fairness :-

Bias Score :- It is used to evaluate and quantify biases present in language models.

Bias score is metric used to evaluate, biases can manifest as preferential treatment of certain groups or perspectives, reinforcing stereotypes, or other forms of unfair representation.

- Evaluating bias is crucial for ensuring fairness and ethical use of language models in various applications.

Some observed biases in training data on LLMs are :-

- Gender Bias
- Racial Bias
- Age Bias
- Socioeconomics Bias

Bias score are often calculated by comparing probabilities or embeddings of biased terms against neutral or unbiased terms.

- Calculations can involve several methods, including word embedding association tests (WEAT),

WEAT :- To measure association b/w target and attribute words.

$$\text{Bias Score} = \frac{1}{N} \sum_{i=1}^{N} \text{Bias Measure}(x_i)$$

Eg :- To evaluate gender bias in Language model using word embeddi ng.

Using WEAT method :- Compares gender-specific and attribute term associations.

Step 1 :- Select target and attribute words :

Target words : (gendered terms) = ["man", "woman"]

Attribute words (career and family) :

Career : ["executive", "management", "professional", "corporation", "salary"]

Family : ["home", "parents", "children", "family", "relatives"]

Step 2 :- Compute Embeddings :

Use a pre-trained model to generate embeddings for target and attribute words.

Step 3: Calculate association:

Compute cosine similarity b/w target and attribute words.

Let's assume:-

- Man-career: [0.85, 0.80, 0.82, 0.78, 0.79]
- Man-family: [0.40, 0.42, 0.38, 0.35, 0.37]
- Woman-career: [0.60, 0.62, 0.65, 0.58, 0.59]
- Woman-family: [0.75, 0.77, 0.72, 0.78, 0.76]

Step 4: Compute bias Measure:-

- Calculate average cosine similarity for each pair and compare differences.

$$\text{Bias} = \left(\frac{1}{5} \sum \text{cosine}(\text{Man-Career}) - \text{cosine}(\text{man-family}) \right) - \left(\frac{1}{5} \sum \text{cosine}(\text{Woman-Career}) - \text{cosine}(\text{woman-family}) \right)$$

$$\text{Bias} = \left(\frac{1}{5}(0.85 + 0.80 + 0.82 + 0.78 + 0.79) - \frac{1}{5}(0.40 + 0.42 + 0.38 + 0.35 + 0.37) \right) - \left(\frac{1}{5}(0.60 + 0.62 + 0.65 + 0.58 + 0.59) + \frac{1}{5}(0.75 + 0.77 + 0.72 + 0.78 + 0.76) \right)$$

$$\text{Bias} = (0.81 - 0.38) - (0.61 - 0.76)$$

$= 0.58$ indicating significant gender bias in model's associations b/w career and family terms.

Fairness Metrics :- It is used to evaluate how equitably a language model treats different groups or individuals.

Bias Score :- quantifies bias that model may have learned during training

Fairness Score : ensure model decisions are fair across diff. groups

Different types of fairness metrices :-

- Demographic Parity : measures whether different demographic groups recieve same outcomes
- Equal Opportunity : evaluates if model provide equal true positive rates for different groups.
- Equalized odds : ensure both true positive and false positive rates are equal across group.
- Calibration : checks if predicted probabilities are accurate across different group.
- Fairness through unawareness : ensures model's decisions don't explicity use protected attributes (like race or gender)
- Individual fairness : ensures similar individuals recieve similar outcomes.

eg :- We want to ensure that model provide equally helpful responses to inquiries from user of different demographics

eg : gender

male	100	80	80%
Non-binary	50	30	60%

$$\text{Fairness Score} = 1 - \left(\frac{\text{max helpful response rate} - \text{min helpful response rate}}{\text{Max helpful response rate}} \right)$$

$$1 - \left(\frac{80-60}{80}\right) = 1 - \frac{20}{80} = 0.75$$

0.75 indicates disparity, model's helpfulness is less equitable across different demographics.

Efficiency Metrics :-

(a) **Latency** :- delay from the moment an input is provided to a language model to moment its generates output. (ms)

Several components of it :-

- **Inference Time** :- time taken by model to process i/p and generate O/p.
- **Network latency** :- time taken for data to travel b/w user & server hosting the model.
- **Pre processing, Post processing** :- time spent on preparing and formatting i/p Output.

Some famous Automated Platforms for LLM evaluation :-

(a) **Shaip.com** : It provides options for online, offline evaluation of LLM model.

(b) **Deepchecks** : most famous, provides solution to address issues like hallucination, incorrect answer, bias, harmful content etc.

Benchmarks for LLM evaluation :-

LLM requires comprehensive benchmarks to measure their performance across various tasks and dimensions, here are some :-

(i) General Language Understanding Evaluation (GLUE) :-

Description : It is widely used benchmark for evaluating the performance of models on natural language understanding (NLU) tasks.

→ It consists of 9 tasks, including sentiment analysis, textual entailment, and sentence similarity [Tasks :- CoLA, SST-2, MRPC, STS-B, QQP, MNLI, QNLI, RTE, WNLI]

Example task :- (a) Sentiment Analysis (SST-2) :- given a sentence, determine whether its sentiment is +ve or -ve.

(MNLI)

(b) Textual Evaluation : given a pair of sentences, determine whether 2nd sentence logically follows from 1st or contradicts it or neutral.

(ii) SuperGLUE :-

Description :- It is extension of GLUE, designed to be more challenging, with additional tasks that require more advanced reasoning and knowledge.

[8 tasks :- BoolQ, CB, CoPA, MultiRC, ReCoRD, RTE, WiC, WSC]

eg :- **BoolQ** : A question-answering task where the model must determine if a given sentence answer a yes/No question based on a passage

(iii) Standford Question-Answering dataset (SQuAD) :-

SQuAD is a reading comprehension dataset where models are required to answer questions based on given passage.

SQuAD 1.1 contains questions with answers directly from text, while SQuAD 2.0 include unanswerable questions.

ex :- Given a paragraph from a wikipedia article; answer specific questions using information from passage.

(iv) Common Sense Reasoning (eg:- Winograd Schema challenge) :-

Description : These benchmark test model's ability to reason about everyday situations and common sense knowledge.

eg: Winograd Schema Challenge :- Given a sentence with a pronoun, model must determine which noun the pronoun refers to, based on common sense

(eg :- "The City Council refused the demonstrators a permit because they feared violence".
· Who feared violence ? → The city council or the demonstrators ?)

(v) LAMBADA :- It is language modeling task where models must predict last word of a sentence, requiring understanding of entire context.

(vi) MMLU (Massive Multitask language Understanding) :-

It evaluates models on broad range of tasks, covering various subjects, including Science, humanities, Social Sciences and more.

(vii) Big Bench (Beyond the Imitation Game) :-

It is collection of tasks designed to test language models on diverse and challenging aspects, including reasoning, creativity and generalization.

eg:- model is presented with a story where a decision-maker chooses between options, then learns outcome.
Task is to access whether model can predict original choice without being swayed by hindsight bias.

(viii) HumanEval :- It is used for evaluating models on code generation tasks. Models are asked to generate correct and functional code given natural language prompts.

eg of task:- write a python function that takes a list of numbers and returns list sorted in ascending order.

(ix) Coco Captioning :- This task asseses a model's ability to generate captions for images, which involves understanding the content and context of image.

(x) TruthfulQA :- This benchmark tests a model's ability to generate truthful and factually accurate answers to questions

eg:- Set of questions designed to trap model into giving false or misleading information.

(xi) Evaluating LLM on Hallucination:

Hallucination tasks measure how often and to what extent models generate information about that is plausible but factually incorrect or entirely fabricated.

(xii) Ethical and Bias Evaluation (REALM) :- These benchmark assess whether models produce biased, harmful or unethical content.

(xiii) Multi-modal tasks (eg:- VQA, Image-text Retrival) :- These tasks Evaluate a model's ability to process and understand multiple modalities eg:- text, image.

Visual Question-Answering (VQA) :- given an image and question about image, model must provide correct answer.

(xiv) HELLASWAG :- It is designed to evaluate model's ability to predict next event or action given a context.

eg:- Given a sentence that describe beginning of an action :-
eg:- ("The man picks up the book") model must choose most plausible continuation from several options.

(xv) ANLI (Adversarial natural language Inference) ⊢ It is a benchmark that tests a model's robustness in natural language inference through adversarially generated examples.

→ The tasks involves determining relationship b/w a pair of sentences (entailment, contradiction, neutral).

(xvi) PIQA (Physical Interaction QA) ⊢

It evaluates a models understanding of physical common sense and its ability to reason about everyday scenarios that involves physical interactions.

ex:- Choose the best method to achieve a physical task, like ⊢ "How to prevent a door from closing" from several options.

(xvii) DROP (Discrete Reasoning over paragraphs) ⊢ It is reading comprehension benchmark that involves discrete reasoning over paragraphs.

→ model must not only understand the text but also perform arithmetic operations, handle dates and count objects.

(xviii) ARC (AI2 Reasoning challenge) ⊢ This benchmark focuses on science questions from grade-school curricula.

It tests model's ability to reason and use scientific knowledge to answer multiple choice questions.

(xix) XGLUE (Cross-lingual General language Understanding Evaluation) :-

It is cross-lingual benchmark for evaluating language understanding tasks across multiple languages.

It includes tasks like machine translation, news classification, question answering in various languages.

eg:- Translate a new headline from english to another language or perform sentiment analysis on a text in multiple languages.

(xx) Semval (Semantic Evaluation) :-

It is on-going series of challenges for evaluating semantic analysis system across a variety of tasks, including sentiment analysis, textual similarity, word sense disambiguation.

eg:- Determine sentiment of tweet (+ve, -ve, neutral) of a tweet.

(xxi) TREX (Textual Retrieval and extraction benchmark) :-

It is a dataset designed for open-domain information extraction and retrieval.

→ Task is to extract and retrieve factual knowledge from large text corpora.

eg:- Given a query like "who is CEO of Apple?"

↑

retrieve correct information from a text corpus.

(xxii) Turing Bench :- TuringBench is a multi-task Benchmark.

→ It evaluates a wide range of language model capabilities, including language generation, understanding and reasoning.

→ It also includes adversial examples.

(xxiii) REALM (Retrieval-Augmented language model) :- It focuses on retrival-based tasks where model must pull relevant information from a large database to answer questions accurately.

(xxiv) Ethical & Bias Benchmark (eg: Fairness in ML) :-

This benchmark evaluate ethical implications of LLMs, such as bias, fairness and production of harmful content.