AI Metrics that Matter: A Guide to Assessing Generative AI Quality

December 3, 2024 | 5 mins

Generative AI models are powerful tools capable of generating content, mimicking human creativity, reasoning, and outputs. These models excel in generating text,

videos, audio, and other innovative outputs which makes the use of these generative models crucial in various fields.

However, assessing the quality of these generative AI models isn't as straight as evaluating traditional AI models. Unlike classification or regression models where accuracy or mean squared error might suffice, generative models produce outputs that are often subjective in nature. The quality of a generated poem, image, or piece of music can't be fully captured by a single numerical metric. Therefore, a combination of quantitative and qualitative metrics is essential to comprehensively evaluate generative AI models.

In this blog we explore the key metrics that matter when assessing the quality of generative AI. By understanding these metrics in detail, one can better fine-tune their models, ensuring that the AI not only performs well on technical benchmarks but also meets the nuanced needs of human users.

Quantitative Metrics

Quantitative metrics are objective, numerical measures used to evaluate specific attributes of a system or process. They provide clear, reproducible, and data-driven evaluations that are typically calculated using mathematical formulas or statistical methods. These metrics are objective, meaning they provide consistent results independent of individual interpretation. Results are expressed in numbers, percentages, or ratios and the metrics can be calculated using algorithms without human intervention.

Perplexity (PPL)

Perplexity (PPL) is a fundamental metric in natural language processing (NLP) which is used to evaluate the performance of language models. It quantifies how well a model predicts a sample of text. It is used to evaluate how well a probabilistic model predicts a sequence of data.

Perplexity serves as a measure of how uncertain or confused a language model is when it tries to predict the next word (or token) in a sentence. Low perplexity means the model is confident and makes good predictions, i.e. it assigns high probabilities to the correct words.

High perplexity means the model is struggling and is less confident, spreading its guesses across many possible words. In simple terms, perplexity tells us how well the model understands language. If the model performs well, it has less "confusion" (low perplexity) and if the model performs poorly, it has more "confusion" (high perplexity).

Perplexity is the exponentiation of the average negative log-likelihood of a sequence. It measures the uncertainty of a model when predicting the next element (e.g., a word or token) in a sequence.

Mathematically, for a sequence $X=(x1,x2,...,xt)$ perplexity is defined as:

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

Where:

- t: Length of the sequence.
- **$p\theta(x_i|x_{<i})$**: The probability assigned by the model to the i-th token xi, given the preceding tokens x<i.

Example:

Suppose we have a sequence of 4 words: **"The cat sat down."**

The model $pH_\theta O$ predicts the conditional probabilities of each word given the previous words as follows:

$pH_\theta O$("The"|Start) = 0.1

$pH_\theta O$("cat"|"The") = 0.2

$pH_\theta O$("sat"|"Thecat") = 0.4

$pH_\theta O$("down"|"Thecatsat") = 0.25

Step 1: Calculate log probabilities for each word in the sequence:

log $pH_\theta O$("The"|Start) = log(0.1) = −2.3026

log $pH_\theta O$("cat"|"The") = log(0.2) = −1.6094

log $pH_\theta O$("sat"|"Thecat") = log(0.4) = −0.9163

log $pH_\theta O$("down"|"Thecatsat") = log(0.25) = −1.3863

Step 2: Compute the average log probability by summing up the log probabilities and divide by the sequence length (t=4):

$$\text{Average log probability} = (-2.3026) + (-1.6094) + (-0.9163) + (-1.3863) / 4$$

$$= -6.2146 / 4$$

$$= -1.5537$$

Step 3: Calculate the Perplexity. It is the exponentiation of the negative average log probability:

$$PPL(x) = \exp(-(-1.5537)) = \exp(1.5537) = 4.73$$

The perplexity for the sequence "The cat sat down." is approximately:

$$PPL(x) = 4.73$$

For the sentence "The cat sat down," a perplexity of 4.73 means the model is as uncertain as if it were picking from 4.73 possible words at each step. Lower perplexity means the model predicts better.

Fréchet Inception Distance (FID)

Fréchet Inception Distance (FID) is a metric used to evaluate the quality of images generated by generative models, particularly Generative Adversarial Networks (GANs). It was introduced by **Martin Heusel et al.** in 2017. FID measures how similar the statistics of generated images are to those of real images. It does this by comparing the means and covariances of feature representations extracted from a pretrained Inception v3 network. Here's a detailed explanation of how FID works, why it's important, and how it's calculated.

Assessing the quality of generated images by generative models, such as GANs, is important. Traditional pixel-wise error metrics like **Mean Squared Error (MSE)** are not sufficient because they don't align well with human perception. Other metrics, such as the **Inception Score (IS)** also had limitations. IS ignores real data distribution as it evaluates only the generated images without considering the real images. IS may also not detect mode collapse, where the generator produces limited diversity. IS also relies on the pretrained Inception network's classification which is not suitable for all datasets. FID addresses these issues by comparing the statistical distributions of real and generated images and provides more effective evaluation.

The Frechet Distance is the mathematical foundation of the FID. The Frechet Distance is a mathematical formula to measure the distance between two multivariate normal distributions. For a simple (univariate) normal distribution, it is calculated as:

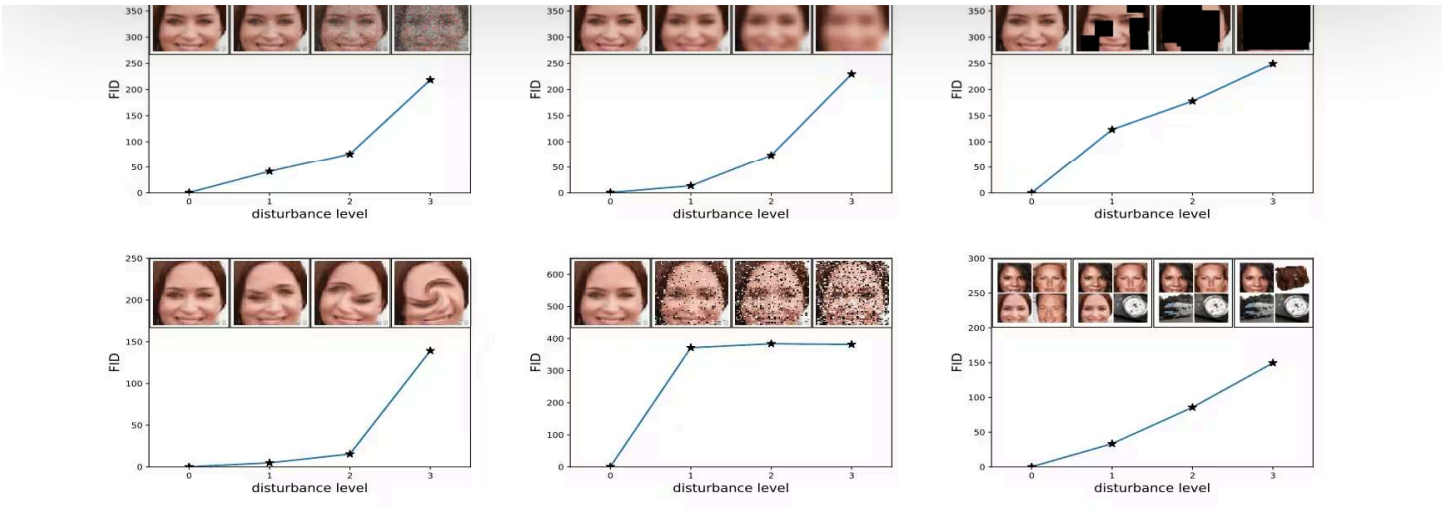
$$d(X,Y) = (\mu_X - \mu_Y)^2 + (\sigma_X - \sigma_Y)^2$$

Here, μX, μY are means of the distributions X and Y and σX, σY are standard deviations of X and Y. The FID is used to evaluate GANs by comparing the similarity between real and generated image distributions. Here's how it works. First, the features are extracted from an intermediate layer of a pre-trained Inception v3 model. These activations (feature vectors) are assumed to follow a multivariate normal distribution. The FID measures how similar the two distributions are in this feature space. For multivariate normal distributions, the FID is given by:

$$FID = ||\mu_X - \mu_Y||^2 - Tr(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y})$$

Here, μX,μY are mean vectors of the real (X) and generated (Y) feature activations.

ΣX,ΣY are **covariance matrices** of the real and generated feature activations. Tr is the **trace of the matrix** (sum of its diagonal elements).



Example of the correlation between increasing image distortion and FID scores (Source)

The figure above demonstrates how the Fréchet Inception Distance (FID) effectively quantifies varying levels of image distortions. The figure presents six types of disturbances (Gaussian noise, Gaussian blur, implanted black rectangles, image swirling, salt and pepper noise, and contamination) of the CelebA dataset with ImageNet images. For each distortion type, the severity increases incrementally from no disturbance to the highest level. As the intensity of these disturbances increases, the FID score also increases which indicates that FID accurately captures and reflects the extent of degradation in image quality.

Bilingual Evaluation Understudy (BLEU)

BLEU (Bilingual Evaluation Understudy) is a metric used to evaluate the quality of text generated by machine translation models or other natural language generation systems. It measures how closely the machine-generated text matches a reference text written by a human.

BLEU is based on the idea of comparing n-grams (sequences of n words) in the generated text to the n-grams in the reference text. The more n-grams that match between the generated and reference texts, the better the score. However, BLEU also considers brevity and penalizes outputs that are too short.

Following are the components of BLUE

N-gram Precision

BLEU evaluates the precision of n-grams (e.g., unigrams, bigrams, trigrams, etc.). The precision is the ratio of overlapping n-grams between the generated text and the reference text to the total number of n-grams in the generated text.

BLEU is typically calculated using multiple n-gram lengths, with common settings being unigram (1-word), bigram (2-word), trigram (3-word), and 4-gram (4-word) precision.

Brevity Penalty

BLEU includes a brevity penalty to discourage models from producing very short outputs, which might have high precision but miss a lot of content. If the generated text is shorter than the reference, a penalty is applied. The penalty ensures that translations are of similar length to the reference, reflecting the idea that translations that are too short might be incomplete.

Geometric Mean

BLEU calculates the geometric mean of n-gram precision scores to account for both local and global alignment. This means that matching higher-order n-grams (like bigrams and trigrams) is more important than just matching individual words (unigrams).

Mathematically BLEU is expressed as:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

Where:

- N is the highest n-gram precision.
- pH_nO is the precision for n-grams of length n.
- BP is the Brevity Penalty applied if the length of the generated text is shorter than the reference text.

The BP is defined as:

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

Let's see an example of calculating BLEU score. Assume following sentences:

Reference Text:	"The cat sat on the mat and watched the birds outside."
Generated Text:	"The cat rested on the mat and stared at the birds."

Step 1: First we tokenize the sentences (split both texts into words).

Reference:	["The", "cat", "sat", "on", "the", "mat", "and", "watched", "the", "birds", "outside"]
Generated:	["The", "cat", "rested", "on", "the", "mat", "and", "stared", "at", "the", "birds"]

Step 2: Then, generate N-grams by creating n-grams for n = 1 (unigrams), n = 2 (bigrams), n = 3 (trigrams), and n = 4 (4-grams).

Reference n-grams:

Unigrams:	["The", "cat", "sat", "on", "the", "mat", "and", "watched", "the", "birds", "outside"]
Bigrams:	["The cat", "cat sat", "sat on", "on the", "the mat", "mat and", "and watched", "watched the", "the birds", "birds outside"]
Trigrams:	["The cat sat", "cat sat on", "sat on the", "on the mat", "the mat and", "mat and watched", "and watched the", "watched the birds", "the birds outside"]
4-grams:	["The cat sat on", "cat sat on the", "sat on the mat", "on the mat and", "the mat and watched", "mat and watched the", "and watched the birds", "watched the birds outside"]

Generated n-grams:

Unigrams:	["The", "cat", "rested", "on", "the", "mat", "and", "stared", "at", "the", "birds"]
Bigrams:	["The cat", "cat rested", "rested on", "on the", "the mat", "mat and", "and stared", "stared at", "at the", "the birds"]
Trigrams:	["The cat rested", "cat rested on", "rested on the", "on the mat", "the mat and", "mat and stared", "and stared at", "stared at the", "at the birds"]
4-grams:	["The cat rested on", "cat rested on the", "rested on the mat", "on the mat and", "the mat and stared", "mat and stared at", "and stared at the", "stared at the birds"]

Step 3: We calculate the precision for each n-gram level by finding matches between the reference and generated n-grams.

Unigram Precision

Matching unigrams:	["The", "cat", "on", "the", "mat", "and", "the", "birds"] (8 matches)
Total unigrams in generated text:	11
Unigram Precision (PH_1O):	8/11 = 0.727

Bigram Precision

Matching bigrams:	["The cat", "on the", "the mat", "mat and", "the birds"] (5 matches)
Total bigrams in generated text:	10
Bigram Precision (PH_2O):	5/10 = 0.5

Trigram Precision

Matching trigrams:	["on the mat", "the mat and"] (2 matches)
Total trigrams in generated text:	9
Trigram Precision (PH_3O):	2/9 = 0.222

4-gram Precision

Matching 4-grams:	["on the mat and"] (1 match)
Total 4-grams in generated text:	8
4-gram Precision (PH_4O):	1/8 = 0.125

Step 4: Calculate the Brevity Penalty (BP)

Length of reference text (r):	11 tokens

Length of generated text (c):	11 tokens
Brevity Penalty (BP):	exp(1−11/11)=1.0

Step 5: Finally, we calculate the BLEU Score. First we calculate logarithms of precisions:

$\log P1 = \log(0.727) = -0.318$

$\log pH_2O = \log(0.5) = -0.693$

$\log pH_3O = \log(0.222) = -1.504$

$\log pH_4O = \log(0.125) = -2.079$

Step 6: Calculate average logarithm:

$= 1/4 \times (-0.318-0.693-1.504-2.079) = -1.149$

Step 7: Compute the exponentiation:

$\exp(-1.149) = 0.316$

Step 8: Apply Brevity Penalty:

$BLEU = 1 \cdot 0.316 = 0.316$

In this example the BLEU score for the generated text is approximately 0.316. This BLEU score indicates that the generated text has moderate similarity to the reference text, with good unigram and bigram matches but fewer matches for higher n-grams. This reflects a typical translation scenario where the general structure is correct but differs in word choice and phrasing.

Rouge (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics commonly used to evaluate the quality of summaries generated by natural language

processing models. It measures how well a generated summary (candidate summary) matches a reference summary by comparing overlapping n-grams, word sequences, or word pairs. ROUGE focuses primarily on recall but can also incorporate precision and F1-score, depending on the variation of the metric used.

ROUGE evaluates the similarity between the generated summary and the reference summary, assuming that high-quality summaries share similar linguistic patterns and content. It is widely used in tasks like text summarization and machine translation. There are many variants of ROUGE metrics as given below:

- ROUGE-N: Measures overlap between n-grams in the prediction and reference.
- ROUGE-L: Uses the Longest Common Subsequence (LCS) to account for word order.
- ROUGE-W: Focuses on weighted LCS, prioritizing consecutive matches.
- ROUGE-S: Evaluates similarity using skip-bigrams (pairs of words with gaps).
- ROUGE-SU: Extends ROUGE-S by including unigram overlaps as well.

Here we will discuss the most common metrics i.e. ROUGE-N.

ROUGE-N

ROUGE-N refers to the direct n-gram overlaps between the candidate (prediction) and the reference. It measures the fraction of n-grams in the reference that appear in the candidate summary (recall) and the fraction of n-grams in the candidate that also appear in the reference (precision). The ROUGE-N score can be expressed in terms of recall, precision, and the F1-score. Common values for N are ROGUE-1 and ROUGE-2.

ROUGE-1 measures unigram (individual word) overlap.

$$\text{ROUGE-1}_{\text{Recall}} = \frac{\text{Unigram}_{\text{cand.}} \cap \text{Unigram}_{\text{ref.}}}{|\text{Unigram}_{\text{ref.}}|}$$

$$\text{ROUGE-1}_{\text{Precision}} = \frac{\text{Unigram}_{\text{cand.}} \cap \text{Unigram}_{\text{ref.}}}{|\text{Unigram}_{\text{cand.}}|}$$

$$\text{ROUGE-1}_{F1} = 2 \times \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

ROUGE-2 measures bigram (two consecutive words) overlap.

$$\text{ROUGE-2}_{\text{Recall}} = \frac{\text{Bigram}_{\text{cand.}} \cap \text{Bigram}_{\text{ref.}}}{|\text{Bigram}_{\text{ref.}}|}$$

$$\text{ROUGE-2}_{\text{Precision}} = \frac{\text{Bigram}_{\text{cand.}} \cap \text{Bigram}_{\text{ref.}}}{|\text{Bigram}_{\text{cand.}}|}$$

$$\text{ROUGE-2}_{F1} = 2 \times \frac{\text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

Let's see an example of how to calculate ROUGE-1.

Reference #1	A fast brown dog jumps over a sleeping fox
Reference #2	A quick brown dog jumps over the fox
Candidate	The quick brown fox jumps over the lazy dog

Step 1: Extract Unigrams

Reference #1 Unigrams:	{"A", "fast", "brown", "dog", "jumps", "over", "a", "sleeping", "fox"} = 9
Reference #2 Unigrams:	{"A", "quick", "brown", "dog", "jumps", "over", "the", "fox"} = 8
Candidate Unigrams:	{"The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"} = 9

Step 2: Find Overlapping Unigrams

Overlapping Unigrams with Reference #1

Reference #1 Unigrams:	{“A”, “fast”, “brown”, “dog”, “jumps”, “over”, “a”, “sleeping”, “fox}

Candidate Unigrams:	{“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”}
Common (Overlapping) Unigrams:	{"brown", "fox", "jumps", "over", "dog"}
Count of overlapping unigrams:	5

Overlapping Unigrams with Reference #2

Reference #2 Unigrams:	{“A”, “quick”, “brown”, “dog”, “jumps”, “over”, “the”, “fox”}
Candidate Unigrams:	{“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”}
Common (Overlapping) Unigrams:	{"quick", "brown", "fox", "jumps", "over", "the", "dog"}
Count of overlapping unigrams:	7

Step 3: Calculating Recall

Recall (Reference #1) = 5/9 = 0.556

Recall (Reference #2) = 7/8 = 0.875

Step 4: Calculating Precision

Precision (Reference #1) = 5/9 = 0.556

Precision (Reference #2) = 7/9 = 0.778

Step 5: F1-Score for Each Reference

F1 (Reference #1) = (0.556x0.556)/(0.556+0.556) = 0.556

F1 (Reference #2) = (0.875x0.778)/(0.875+0.778) = 0.823

Step 6: Final ROUGE-1 Scores (Average Across References)

Average Recall = 0.556 + 0.875/2 = **0.716 (71.6%)**

Average Precision = 0.556 + 0.778/2 = **0.667 (66.7%)**

Average F1 Score = 0.556 + 0.823/2 = **0.690 (69.0%)**

Inception Score (IS)

The **Inception Score (IS)** is a widely used metric for evaluating the performance of generative models for the image generation tasks. The metric was introduced as part of evaluating Generative Adversarial Networks (GANs) but has since been applied to a variety of generative models. The Inception Score helps measure two critical aspects of generated images that are image quality and image diversity. For image quality it measures how realistic and interpretable the generated images are and for image diversity it measures how varied the generated images are. This makes sure that generated images are sharp and contain clear objects and the model generates a wide range of outputs rather than similar or repetitive images. It uses a pre-trained Inception v3 Network to measure two desirable properties of a generative model.

The Inception Score is mathematically defined as:

$$\mathrm{IS}(G) = \exp\left(\mathbb{E}_{\mathbf{x}\sim p_g} \, D_{KL}(\, p(y|\mathbf{x}) \parallel p(y) \,) \right)$$

Where,

- X~pH$_g$O: x is a generated image sampled from the generative model's distribution pH$_g$O.
- p(y|x): The conditional class distribution predicted by the Inception model for image x. It indicates how confident the model is about classifying the image into a particular class.

- p(y): The marginal class distribution, computed as

$$p(y) = \int_{\mathbf{x}} p(y|\mathbf{x}) p_g(\mathbf{x})$$

- $DH_{KL}O$: The Kullback-Leibler (KL) divergence, which measures how different two probability distributions are.

Let's understand IS through an example. Let's create a simplified example of the Inception Score calculation based on the example **here**. Instead of using the full CIFAR-10 dataset and the InceptionV3 model, we'll simulate the process using small sample data and simple calculations.

Step 1: Define Class Probabilities p(y|x)

These are the probabilities assigned to each class for 3 generated images. Each row represents the probabilities for one image, and each column corresponds to a class (3 classes in total).

Image 1: $p(y|xH_1O)$=[0.7, 0.2, 0.1]

mage 2: $p(y|xH_2O)$=[0.1, 0.8, 0.1]

Image 3: $p(y|xH_3O)$=[0.2, 0.2, 0.6]

Step 2: Calculate the Marginal Class Distribution p(y)

The marginal distribution p(y) is the mean of p(y|x) across all images, giving us the overall probability of each class.

1/3 x ([0.7, 0.2, 0.1] + [0.1, 0.8, 0.1] + [0.2, 0.2, 0.6]) = [0.333, 0.4, 0.267]

Step 3: Calculate KL Divergence for Each Image

The KL Divergence for each image measures how different the predicted p(y|x) is from the marginal distribution p(y). For each image, we calculate:

Image 1: $DH_{KL}O(p(y|x1)\|p(y)) =$

$(0.7 \log 0.7/0.333)+(0.2 \log 0.2/0.4)+(0.1 \log 0.1/0.267) = 0.362$

Image 2: $DH_{KL}O(p(y|x1)\|p(y)) =$

0.456

Image 3: $DH_{KL}O(p(y|x1)\|p(y)) =$

0.261

Step 4: Average KL Divergence

We average the KL divergence over all images:

$AVGH_{KL}O = 1/3 \times (0.362+0.456+0.261) = 0.36$

Step 5: Calculate the Inception Score

The Inception Score is the exponential of the average KL divergence:

$IS= \exp(0.36) = 1.43$

In this example, the Inception Score is approximately 1.43. Higher scores indicate that the generated images are both high-quality and diverse.

METEOR

The **METEOR** (Metric for Evaluation of Translation with Explicit ORdering) metric is used to evaluate various NLP tasks, such as machine translation, summarization, and other generative AI applications. It addresses some of the limitations of earlier metrics like BLEU, focusing on a more comprehensive alignment between generated and reference texts. Key Components of METEOR are:

Alignment

METEOR creates alignments between the candidate (system) translation and the reference translation by mapping unigrams (individual words) through various stages:

1. Exact Matching: Direct word matches.
2. Stemming: Matches based on word stems using tools like the Porter stemmer.
3. Synonymy: Matches based on synonyms, often utilizing resources like WordNet.

Each stage attempts to map unigrams not previously aligned, ensuring that each word in one string maps to at most one word in the other.

Crossing Penalty

To maintain the order of words, METEOR minimizes "crossings" in alignments. A crossing occurs when the positional order of mapped words between the candidate and reference translations is inconsistent. The metric selects alignments with the fewest such crossings to better reflect the correct word order.

Precision and Recall

Precision is the ratio of matched unigrams to the total unigrams in the candidate translation and recall is the ratio of matched unigrams to the total unigrams in the reference translation. METEOR combines these using a harmonic mean, with recall typically weighted higher than precision to better reflect human judgment.

$$FMean = \frac{10PR}{R + 9P}$$

Fragmentation Penalty

For the word order, METEOR identifies chunks of contiguous matches. A penalty is applied based on the number of chunks. If there are more chunks, it indicates higher fragmentation and resulting in a lower score.

$$Penalty = 0.5 \times \frac{\#\text{ of Chunks}}{\#\text{ of Unigrams Matched}}$$

Final Score

The **final METEOR score** adjusts the harmonic mean of precision and recall by applying the fragmentation (word order) penalty, providing a comprehensive measure of translation quality that accounts for both content and word order.

$$\text{METEOR} = \underbrace{\text{FMean}}_{\text{Harmonic Mean of Unigram Precision/Recall}} * \underbrace{(1 - \text{Penalty})}_{\text{Word Order Penalty}}$$

We will see an example of METEOR score calculations. Consider the following sentences:

Reference:	"The quick brown fox jumps over the lazy dog."
Candidate:	"A fast brown fox leaps over a lazy dog."

Matching details are:

Exact Matches:	"brown," "fox," "over," "lazy," "dog."
Synonym Matches:	"quick" ↔ "fast," "jumps" ↔ "leaps."

Calculating the scores:

Precision (P):	7 matches / 9 candidate words = 0.778
Recall (R):	7 matches / 9 reference words = 0.778
F-mean:	(10×0.778×0.778) / (0.778+9×0.778) = 0.778
Chunks (c):	2 (e.g., "fast brown fox" and "over a lazy dog")
Matched Unigrams:	7
Penalty:	0.5×(2/7)H^3O = 0.0058
METEOR Score:	0.778×(1−0.0058) = 0.773

This process results in a METEOR score that reflects both the accuracy and fluency of the candidate translation, aligning closely with human judgment.

Qualitative Metrics

Qualitative metrics are subjective assessments that evaluate the quality of outputs based on human interpretation, judgment, or experiential feedback. These metrics are particularly useful for capturing aspects of generative AI outputs that are difficult to quantify, such as creativity. These metrics are subjective, meaning results may vary based on human evaluators' perspectives. It is often expressed through ratings, feedback, or qualitative analysis. It depends on the task, audience, and domain-specific expectations.

Human Evaluation

Human evaluation involves human judges assessing the outputs generated by GenAI models based on predefined criteria like fluency, creativity, or relevance. This method is important because GenAI often generates outputs that are highly context-dependent and subjective.

In this method human evaluators rate the output of the model based on specific metrics like clarity, accuracy etc. A pairwise comparisons or ranking methods are also commonly used, where evaluators compare two or more model outputs.

For example evaluating a chatbot's responses in a conversational AI model, a human evaluator judges responses to prompts like "Tell me about the benefits of solar energy.", the evaluation criteria may be used such as

- Relevance: Does the response focus on solar energy benefits?
- Fluency: Is the response grammatically correct and natural-sounding?
- Engagement: Is the tone friendly or engaging enough for a user?

Creativity and Novelty Metrics

Creativity and novelty in GenAI evaluate how original or innovative the generated outputs are. These metrics measure the model's ability to produce unique content that is not just a rephrasing of training data. Novelty can be measured by comparing generated outputs against the training data using similarity measures. Human judges or domain experts evaluate creativity, particularly when outputs like stories, art, or poems are subjective and context-specific.

For example, evaluating a GenAI model for writing fictional short stories human judges may use a prompt "Write a story about a futuristic city powered entirely by AI." and uses following criteria for evaluation:

- Creativity: Does the story introduce innovative ideas?
- Novelty: Is the story significantly different from existing sci-fi examples from the training data?

Coherence and Consistency

Coherence ensures the generated text is logically structured and flows well. On the other hand, consistency checks whether the details (e.g., character names, context, tone) remain uniform throughout the generated output. In this evaluation process models are tested on long outputs where maintaining coherence and consistency is harder (e.g., multi-paragraph stories or conversations).

For example, evaluating the GenAI model for generating research papers a human judges may issue a prompt like "Write a research abstract about quantum computing applications in medicine." and use following evaluation criteria:

- Coherence: Does the abstract present a logical flow of ideas from problem statement to proposed solution?
- Consistency: Are technical terms and methodologies used consistently, without contradictions?

Relevance and Appropriateness

Relevance measures how well the output aligns with the input prompt or task, while appropriateness measures the tone, style, or contextual suitability of the generated content.

Automated metrics like BLEU or ROUGE may evaluate relevance but often fail for tasks requiring contextual understanding. Human evaluators check appropriateness in sensitive applications, such as conversational AI or content creation, to ensure outputs are contextually suitable.

For example, in generating email drafts for business communication a human judge may use prompts such as "Write an email to a client apologizing for a delay in delivery." and use following evaluation Criteria:

- Relevance: Does the email address the delay and include a clear apology?
- Appropriateness: Is the tone professional and empathetic, without being overly casual or robotic?

Key Takeaways: Generative AI Metrics

As GenAI models are used across diverse domains, evaluating their performance requires metrics that balance efficiency, accuracy, and real-world relevance. Below are a few key points regarding AI evaluation metrics:

- Combining quantitative and qualitative metrics provides a balanced assessment of model performance, capturing both measurable outcomes and user satisfaction.
- Metrics should align with the specific goals of the application, whether it is image generation, language modeling, or decision-making tasks.
- Evaluation should include checks for fairness, inclusivity, and bias to ensure responsible AI deployment.
- Metrics should reflect the practical utility of the model, assessing how well it performs under realistic conditions and diverse inputs.

WRITTEN BY

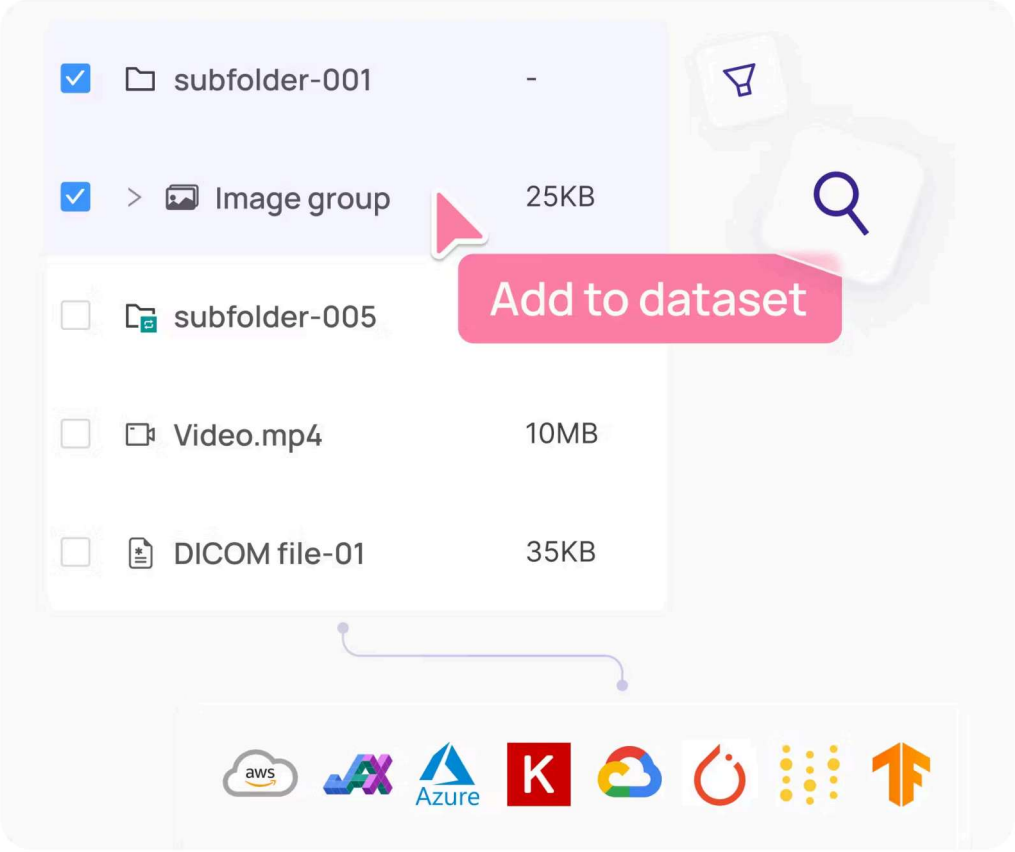
Alexandre Bonnet

View more posts →

PREVIOUS BLOG
How to Label and Analyze Multimodal

NEXT BLOG
How to Manage Data Annotation Pipelines...

Explore our products



Index

Manage & curate your data

Understand and manage your visual data, prioritize data for labeling, and initiate active learning pipelines.

Explore Index →

Your work email

Platform	Modalities	Resources	Company
Data management	Multimodal	Blog	Pricing
Data annotation	Image	Webinars	Customers
Model evaluation	Video	Security	About
Agents	Document & text	Documentation	Careers
	Audio	Learn	Press
	DICOM & NIfTI	Glossary	Contact Us
		AI Data Readiness Report