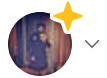




Search Medium

Write



★ Member-only story

UNLOCKING NLP EVALUATION

# Demystifying BLEU, ROUGE, and METEOR: Key Metrics for Evaluating Generated Text in NLP



Armin Norouzi, Ph.D · [Follow](#)

Published in Level Up Coding · 7 min read · Aug 10



168



Unravel the intricacies of BLEU, ROUGE, and METEOR metrics for NLP text quality. Implement and optimize with Python.

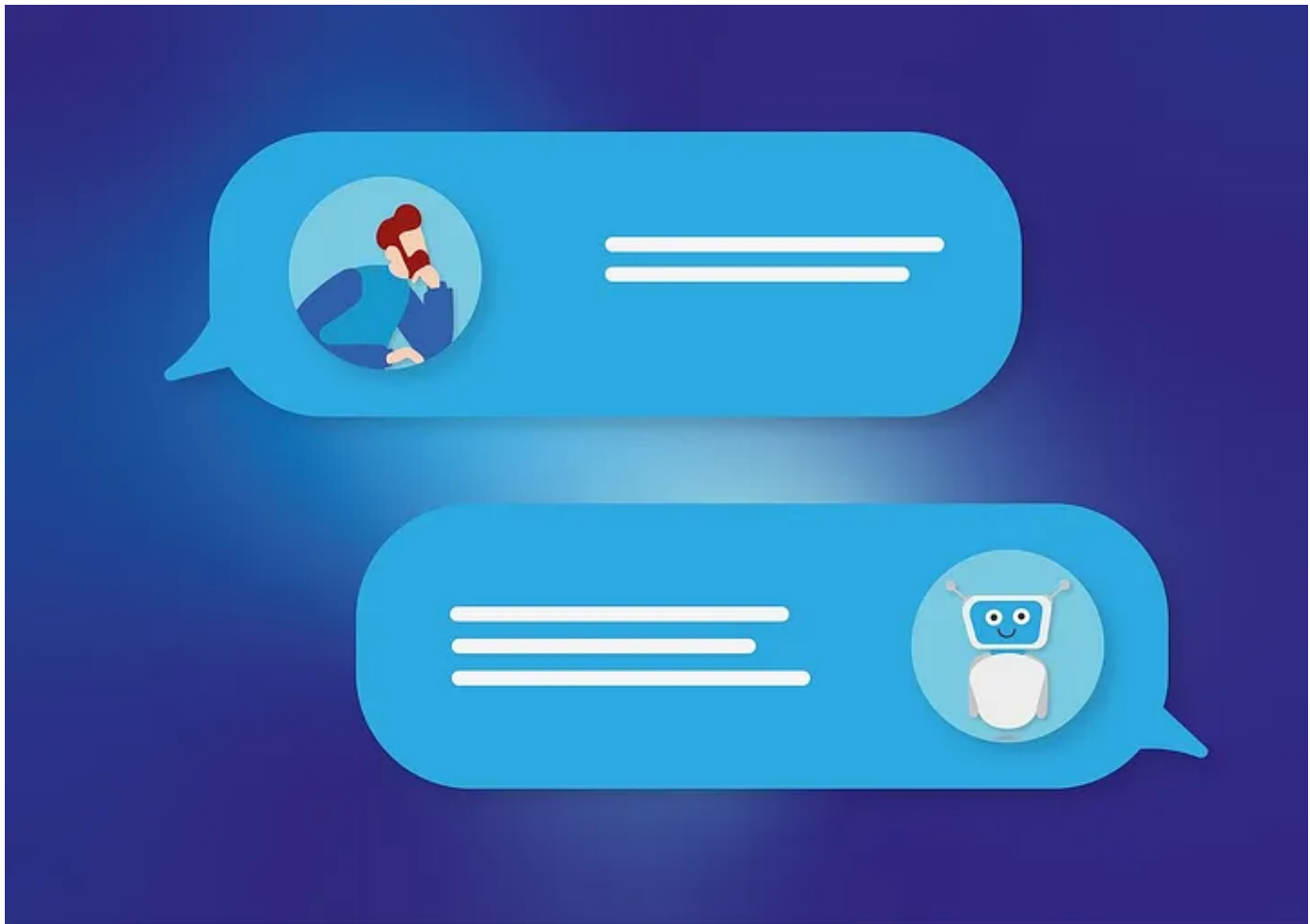




Image from [Pixabay](#).

Before starting, if you want to learn more about data structure, algorithms, data science, and generative AI, I suggest checking out my other posts using the below lists:

Armin Norouzi, Ph.D

## Data Structure and Algorithms Series

[View list](#) 9 stories



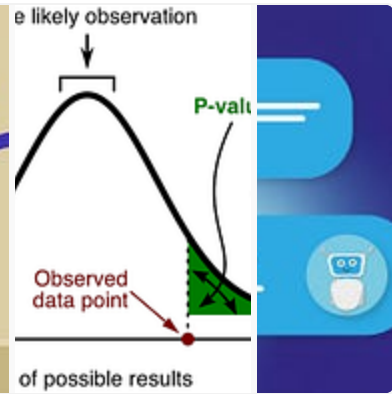


Armin Norouzi, Ph.D

## Machine learning and data science

View list

7 stories

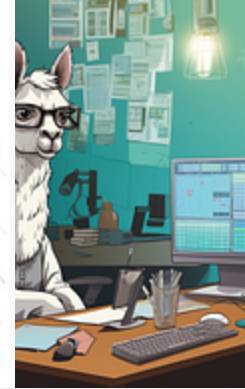
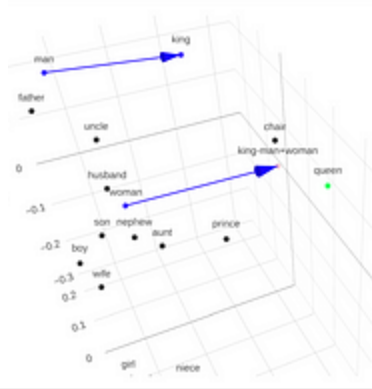


Armin Norouzi, Ph.D

## Generative AI

View list

15 stories



Now, let's get started!

## Introduction

When it comes to assessing the quality of generated text in natural language processing (NLP) tasks, several metrics have emerged to assist researchers and developers in quantifying the performance of their models. These metrics offer valuable insights into the level of similarity between the generated text and the reference text. In this article, we will delve into three widely used generated text evaluation metrics: BLEU, ROUGE, and METEOR. Understanding how these metrics work and implementing them using Python will empower you to improve your NLP applications.

## BLEU (Bilingual Evaluation Understudy)

## Understanding BLEU and its Origin

BLEU, introduced by Kishore Papineni and his colleagues in 2002, was initially developed for machine translation tasks. Since then, it has become a standard metric for evaluating generated text across various NLP domains. BLEU measures the similarity between the generated text and one or more reference texts by calculating the n-gram overlap.

## How BLEU Measures Text Similarity

The BLEU score is based on the precision of n-grams (contiguous sequences of words) present in the generated text compared to the reference text. It rewards the generated text for having n-grams that are also present in the reference text.

## Calculating BLEU Score

Let's imagine a model generated "A fast brown fox jumps over a lazy dog." while we expected a reference Text as: "The quick brown fox jumps over the lazy dog."

**Step 1:** Calculate the precision for each n-gram (unigram, bigram, etc.) in the generated text compared to the reference text.

- Unigrams in generated text: ["A", "fast", "brown", "fox", "jumps", "over", "a", "lazy", "dog", "."]
- Unigrams in reference text: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog", "."]

$Precision = \text{Number of common unigrams} / \text{Total number of unigrams in the generated text}$

$Precision = 9 / 10 = 0.9$

**Step 2:** Calculate the brevity penalty to account for shorter generated texts.

$$\text{Brevity Penalty} = \text{Minimum}(1, (\text{Number of words in generated text} / \text{Number of words in reference text}))$$

So the Brevity Penalty will be a minimum of 1 and 9/10 which will be 0.9

**Step 3:** Calculate the BLEU score.

- $\text{BLEU Score} = \text{Brevity Penalty} * \exp(\text{sum}(\log(\text{precision})))$
- $\text{BLEU Score} = 0.9 * \exp(\log(0.9)) \approx 0.9 * 0.9 \approx 0.81$

The BLEU score for this example is approximately 0.81 (81%).

## Advantages and Limitations of BLEU

One of the key advantages of BLEU is its simplicity and ease of computation. However, it has certain limitations, such as not considering word order or semantics. Long sentences can be penalized due to n-gram precision, and it might not correlate well with human judgments.

## Use Cases of BLEU in Natural Language Processing (NLP)

BLEU is extensively used in machine translation tasks, text summarization, and language generation. Its effectiveness in evaluating text quality makes it a valuable tool in the NLP community.

## ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

## Introduction to ROUGE and its Purpose

ROUGE, developed by Chin-Yew Lin and his team in 2004, was designed specifically for text summarization evaluation. Unlike BLEU, which focuses on precision, ROUGE emphasizes recall, measuring how much of the reference text's information is captured in the generated text.

## How ROUGE Evaluates Text Quality

ROUGE-N evaluates the n-gram overlap between the generated text and the reference text, similar to BLEU. ROUGE-L measures the longest common subsequence, considering word order. ROUGE-W assesses weighted word overlap, giving more weight to specific words or phrases. ROUGE-N, ROUGE-L, and ROUGE-W provide different perspectives on the generated text's quality. By using multiple ROUGE measures, a comprehensive evaluation of the text can be achieved.

## Application of ROUGE in Text Summarization and Machine Translation

ROUGE has gained popularity in evaluating text summarization systems, where it effectively measures the quality of generated summaries. Additionally, it has found applications in machine translation evaluations.

## Calculating ROUGE Score

Let's use the same example as above and go over ROUGE calculations.

**Step 1:** Calculate the overlap of unigrams (ROUGE-1) and bigrams (ROUGE-2) between the generated and reference text.

- Unigrams overlap: 9 ("fox", "jumps", "over", "brown", "lazy", "dog", ":", "a", "brown")
- Bigrams overlap: 5 ("fox jumps", "jumps over", "brown fox", "lazy dog", "a lazy")

**Step 2: Calculate the recall for each ROUGE measure.**

*ROUGE-1 Recall* = Number of overlapping unigrams / Total number of unigrams in the reference text

$$\text{ROUGE-1 Recall} = 9 / 10 = 0.9$$

*ROUGE-2 Recall* = Number of overlapping bigrams / Total number of bigrams in the reference text

$$\text{ROUGE-2 Recall} = 5 / 9 \approx 0.556$$

**Step 3: Calculate the ROUGE score.**

$$\text{ROUGE Score} = (\text{ROUGE-1 Recall} + \text{ROUGE-2 Recall}) / 2$$

$$\text{ROUGE Score} = (0.9 + 0.556) / 2 \approx 0.728$$

The ROUGE score for this example is approximately 0.728 (72.8%).

## **METEOR (Metric for Evaluation of Translation with Explicit Ordering)**

### **Overview of METEOR and its Goals**

METEOR, introduced by Alon Lavie and Abhaya Agarwal in 2005, is another metric for evaluating machine translation. It aims to address some of the limitations of BLEU and ROUGE by incorporating synonyms and paraphrases into the evaluation process.

## METEOR's Approach to Text Evaluation

METEOR combines exact word matches, stemmed word matches, and paraphrase matching. It computes the harmonic mean of these matching scores to determine the overall quality of the generated text.

### Calculating METEOR Score

**Step 1:** Calculate the exact word matches between the generated and reference text.

*Exact Word Matches* = 9 ("fox", "jumps", "over", "the", "lazy", "dog", ".", "brown", "a")

**Step 2:** Calculate the stemmed word matches.

*Stemmed Word Matches* = 1 ("fast")

**Step 3:** Calculate the alignment score.

*Alignment Score* = (Exact Word Matches + Stemmed Word Matches) / Total Words in the generated text

*Alignment Score* = (9 + 1) / 10 = 0.1

**Step 4:** Calculate the METEOR score.

*METEOR Score* = (1 - (Weight \* (1 - Alignment Score))) \* Meteor Penalty

By assuming Weight = 0.85 and Meteor Penalty = 0.775 (for this example):

*METEOR Score* = (1 - (0.85 \* (1 - 0.1))) \* 0.775 = (1 - 0.135) \* 0.775 ≈ 0.66



The METEOR score for this example is approximately 0.66 (66%).

*Please note that the values used for Weight and Meteor Penalty are just examples. In real-world scenarios, these values might vary depending on the specific implementation. The manual calculations provide an understanding of how these metrics work to evaluate generated text.*

## **Incorporating Synonyms and Paraphrases**

One of the unique features of METEOR is its ability to consider synonyms and paraphrases, making it more robust in evaluating text generated by models with linguistic variations.

## **METEOR in Automated Content Generation and Text Rewriting**

METEOR's adaptability has led to its adoption in automated content generation tasks and text rewriting applications, where evaluating paraphrased text is crucial.

## **Comparing BLEU, ROUGE, and METEOR**

### **Differences in Evaluation Methodologies**

BLEU, ROUGE, and METEOR approach text evaluation differently. BLEU focuses on n-gram precision, ROUGE on recall with various measures, and METEOR incorporates synonyms and paraphrases.

### **Which Metric to Choose and When?**

The choice of metric depends on the specific NLP task and the desired evaluation aspect. **BLEU may be more suitable for machine translation,**

while ROUGE could be preferable for text summarization. METEOR excels in tasks with diverse expressions.

## Addressing the Challenges of Automated Evaluation

While these metrics offer valuable insights into text quality, they still cannot completely replace human evaluation. Researchers and developers must acknowledge the challenges and limitations of automated evaluation metrics.

## Implementing BLEU, ROUGE, and METEOR in Python

### Preparing Text Data for Evaluation

Before applying the metrics, it is essential to preprocess the text data, such as tokenization and stemming, to ensure accurate evaluation.

### Step-by-step Python Implementation of BLEU

Let's dive into a Python implementation of the BLEU metric using the NLTK library, making it easier for you to evaluate your generated text.

```
from nltk.translate.bleu_score import sentence_bleu

# Sample reference and generated sentences
reference = ["A", "fast", "brown", "fox", "jumps", "over", "a",
"lazy", "dog", "."]
generated = ["The", "quick", "brown", "fox", "jumps", "over", "the",
"lazy", "dog", "."]

# Calculate BLEU score
bleu_score = sentence_bleu(reference, generated)
print('BLEU Score:', bleu_score)
```

## ROUGE Implementation in Python with NLTK

Similarly, we can implement ROUGE using the NLTK library to evaluate text summarization models.

```
from nltk.translate.bleu_score import corpus_rouge

# Sample reference and generated summaries
reference_summaries = [['A fast brown fox jumps over a lazy dog.']]
generated_summaries = [['The quick brown fox jumps over the lazy dog.']]

# Calculate ROUGE score
rouge_score = corpus_rouge(reference_summaries, generated_summaries)
print('ROUGE Score:', rouge_score)
```

## Calculating METEOR Score using Python Packages

METEOR implementation requires a slightly different approach. We can use existing Python packages designed for METEOR evaluation.

```
from meteor import meteor_score

# Sample reference and generated sentences
reference_sentence = 'A fast brown fox jumps over a lazy dog.'
generated_sentence = 'The quick brown fox jumps over the lazy dog.'

# Calculate METEOR score
meteor_score = meteor_score.meteor_score([reference_sentence], generated_sentence)
print('METEOR Score:', meteor_score)
```

By following these Python implementations, you can effortlessly evaluate the performance of your NLP models using BLEU, ROUGE, and METEOR metrics.

## Conclusions

In this article, we have explored the significance of generated text evaluation metrics: BLEU, ROUGE, and METEOR. Understanding these metrics is crucial for assessing the quality of generated text in various NLP tasks. By implementing these metrics using Python, you can improve your NLP models' performance and deliver more accurate and contextually appropriate text.

Thank you for reading my post, and I hope it was useful for you. If you enjoyed the article and would like to show your support, please consider taking the following actions:



Give the story a round of applause (clap) to help it gain visibility.



Follow me on Medium to access more of the content on my profile.

### Follow Now



Subscribe to the newsletter to not miss my latest posts: **[Subscribe Now](#)** or **[become a referred Medium member](#)**.




Connect with me on **[LinkedIn](#)** for updates.

[AI](#)[Metrics And Analytics](#)[Machine Learning](#)[NLP](#)[Language Model](#)


More from the list: "NLP"

Curated by Himanshu Birla

 Jon Gi... in Towards Data ...


**Characteristics of Word Embeddings**

★ · 11 min read · Sep 4, 2021

 Jon Gi... in Towards Data ...

**The Word2vec Hyperparameters**

★ · 6 min read · Sep 3, 2021

 Jon Gi... in

**The Word2vec**

★ · 15 min read

[View list](#)



Written by Armin Norouzi, Ph.D

322 Followers · Writer for Level Up Coding

Machine Learning Engineer | Data Scientist

Follow

More from Armin Norouzi, Ph.D and Level Up Coding



	Comment	Date
	WIP	3 days ago
	Off for lunch	1 day ago
	End of code for today	20 hours ago
	I am tired AF	18 hours ago
	Happy Weekend Team	16 hours ago
	First to commit	14 hours ago
	Fixed final bug	10 hours ago
	Added a new feature	9 hours ago
	Fixed another bug	7 hours ago
	Made some changes	5 hours ago
	fixed two build-breaking issues	3 hours ago
	Bugs are never ending. fixed another bug 🤔	2 hours ago



Armin Norouzi, Ph.D in Artificial Corner

## Mastering Llama 2: A Comprehensive Guide to Fine-...

Dive deep into Llama 2—the cutting-edge NLP model. This guide covers everything...

★ · 30 min read · Sep 4



258



7



Victor Timi in Level Up Coding

## “Good Commit” vs “Your Commit”: How to Write a Perfect Git Commi...

A good commit shows whether a developer is a good collaborator—Peter Hutterer, Linux.

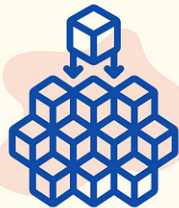
★ · 8 min read · Sep 5



2.6K



32



## 12 Microservices Patterns I Wish I Knew Before the Interview



Arslan Ahmad in Level Up Coding

## 12 Microservices Patterns I Wish I Knew Before the System Design...

Mastering the Art of Scalable and Resilient Systems with Essential Microservices Design...

13 min read · May 16



3.8K



17



Armin Norouzi, Ph.D in Artificial Corner

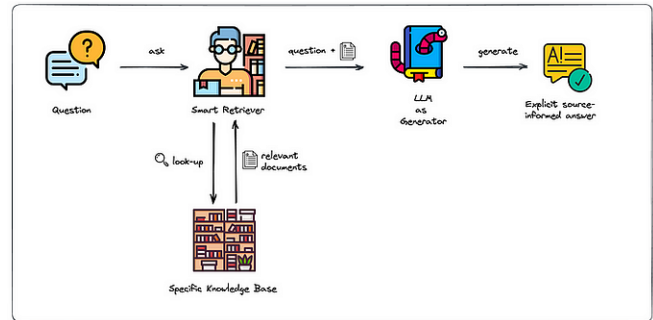
## Retrieval-Augmented Generation (RAG): A Short Introduction

Dive into the essence of RAG in AI: a tool amplifying data richness & optimizing...

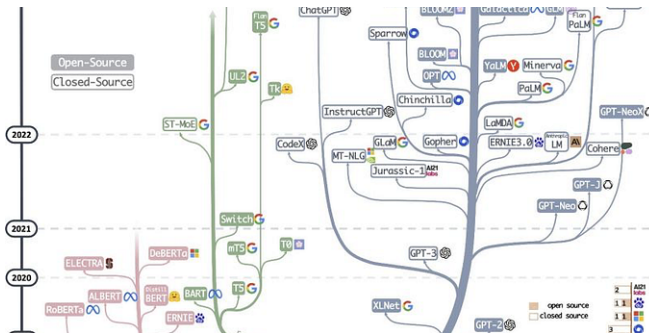
★ · 4 min read · Sep 12



87

[See all from Armin Norouzi, Ph.D](#)[See all from Level Up Coding](#)

## Recommended from Medium



 Haifeng Li

### A Tutorial on LLM

Generative artificial intelligence (GenAI), especially ChatGPT, captures everyone's...

15 min read · Sep 14



372



 Dixn Jakindah

### Top P, Temperature and Other Parameters

Large Language Models (LLMs) are essential tools in natural language processing (NLP)...

3 min read · May 18



## Lists



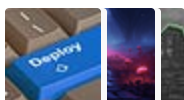
### Natural Language Processing

669 stories · 283 saves



### The New Chatbots: ChatGPT, Bard, and Beyond

13 stories · 133 saves



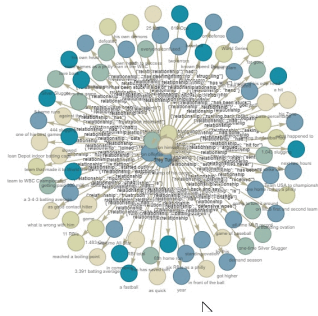
### Predictive Modeling w/ Python

20 stories · 452 saves



### Practical Guides to Machine Learning

10 stories · 519 saves



Wenqi Glantz in Better Programming

## 7 Query Strategies for Navigating Knowledge Graphs With...

Exploring NebulaGraph RAG Pipeline with the Philadelphia Phillies

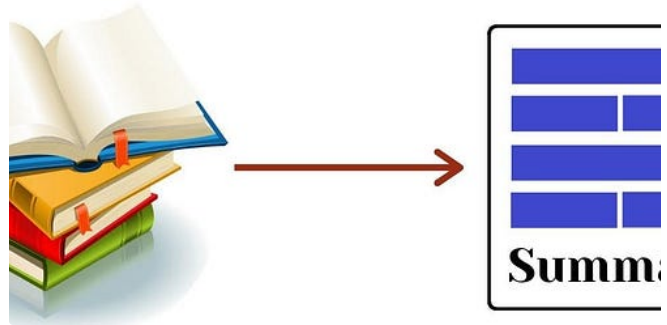
🌟 · 17 min read · 4 days ago



4



...



Fabiano Falcão

## Metrics for evaluating summarization of texts performe...

Text summarization performed by Transformers is one of the most fascinating...

7 min read · Apr 23



4



...



Eren Kızılırmak

## Text Summarization: How To Calculate Rouge Score

This article is written by Eren Kızılırmak and Alparslan Mesri.

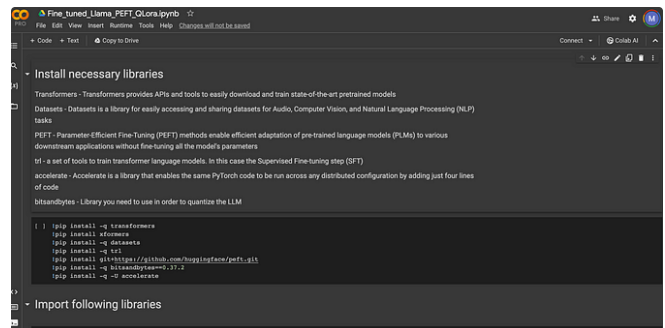
6 min read · Aug 2



4



...



Maya Akim

## Complete Guide to LLM Fine Tuning for Beginners

Fine-tuning a model refers to the process of adapting a pre-trained, foundational model...

5 min read · Aug 14



1



...



See more recommendations