Quantitative Metrics Simplified for Language Model Evaluation

ADVANCED NLP PYTHON

Introduction

Language models are usually trained on extensive amounts of textual data. These models aid in generating natural-sounding responses like humans. Additionally, they can perform various language-related tasks such as translation, text summarization, text generation, answering specific questions, and more. Language models' evaluation is crucial to validate their performance, quality and to ensure the production of top-notch text. This is particularly significant for applications where the generated text influences decision-making or furnishes information to users.

There are various ways to evaluate language models such as human evaluation, feedback from end-users, LLM-based evaluation, academic benchmarks (like GLUE and SQuAD), and standard quantitative metrics. In this article, we will delve deeply into various standard quantitative metrics such as BLEU, ROUGE, and METEOR. Quantitative metrics in the field of NLP have been pivotal in understanding language models and their functionalities. From precision and recall to BLEU and ROUGE scores, these metrics offer a quantitative metrics evaluation of model effectiveness. Let's delve into each traditional metric.

Learning Objectives

- Explore various types of standard quantitative metrics.
- Understand the intuition, and math behind each metric.
- Explore the limitations, and key features of each metric.

This article was published as a part of the Data Science Blogathon.

Table of contents

What is BLEU Score ?

BLEU (BiLingual Evaluation Understudy) score is a metric for automatically evaluating machine-translated text. It evaluates how closely the machine-translated text aligns with a collection of high-quality reference translations. The BLEU score ranges from 0 to 1, with 0 indicating no overlap between the machine-translated output and the reference translation (i.e. low-quality translation), and 1 indicating perfect overlap with the reference translations (i.e. high-quality translation). It is an easy-to-understand and inexpensive-to-compute measure. Mathematically BLEU score is defined as:

$$BLEU_{score} = \ Brevity\ Penalty * Geometric\ Average\ Precision$$

BLEU score calculation

The BLEU score is calculated by comparing the n-grams in the machine-translated text to those in the reference text. N-grams refer to sequences of words, where “n” indicates the number of words in the sequence.

Let’s understand the BLEU score calculation using the following example:

Candidate sentence: They cancelled the match because it was raining.

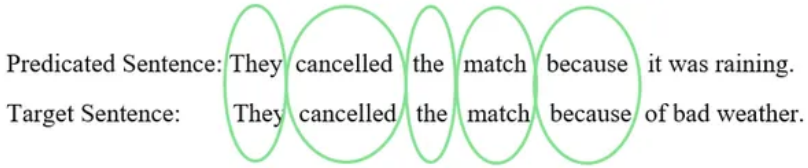
Target sentence: They cancelled the match because of bad weather.

Here, the candidate sentence represents the sentence predicted by the language model and the target sentence represents the reference sentence. To compute geometric average precision let’s first understand the precision scores from 1-gram to 4-grams.

Precision 1-gram

$$Precision\ 1\_gram = \frac{Number\ of\ correctly\ predictaed\ 1\_grams}{Total\ number\ of\ 1\_grams\ in\ the\ predicated\ sentence}$$

Predicated sentence 1-grams: [‘They’, ‘cancelled’, ‘the’, ‘match’, ‘because’, ‘it’, ‘was’, ‘raining’]

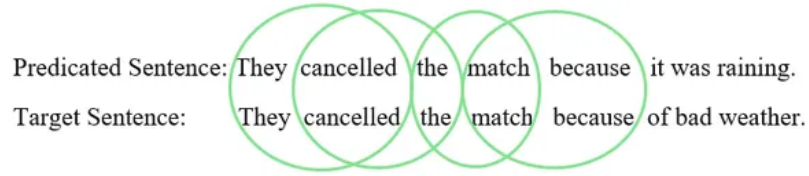


Precision 1-gram = 5/8 = 0.625

Precision 2-gram

$$Precision\ 2\_gram = \frac{Number\ of\ correctly\ predictaed\ 2\_grams}{Total\ number\ of\ 2\_grams\ in\ the\ predicated\ sentence}$$

Predicated sentence 2-grams: [‘They cancelled’, ‘cancelled the’, ‘the match’, ‘match because’, ‘because it’, ‘it was’, ‘was raining’]

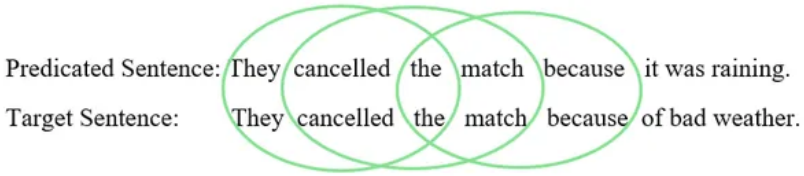


Precision 2-gram = 4/7 = 0.5714

Precision 3-gram

$$Precision\ 3\_gram = \frac{Number\ of\ correctly\ predictaed\ 3\_grams}{Total\ number\ of\ 3\_grams\ in\ the\ predicated\ sentence}$$

Predicated sentence 3-grams: [‘They cancelled the’, ‘cancelled the match’, ‘the match because’, ‘match because it’, ‘because it was’, ‘it was raining’]

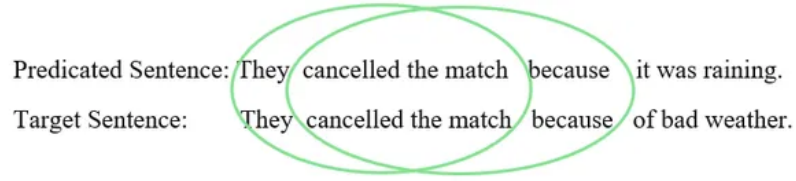


Precision 3-gram = 3/6 = 0.5

Precision 4-gram

$$Precision\ 4\_gram = \frac{Number\ of\ correctly\ predictaed\ 4\_grams}{Total\ number\ of\ 4\_grams\ in\ the\ predicated\ sentence}$$

Predicated sentence 4-grams: [‘They cancelled the match’, ‘cancelled the match because’, ‘the match because it’, ‘match because it was’, ‘because it was raining’]



Precision 4-gram = 2/5 = 0.4

Geometric Average Precision

Geometric average precision with different weights for different n-grams can be computed as

$$Geometric\ Average\ Precision\ (N) = \exp(\sum_{n=1}^{N} w_n\ logp_n) = \prod_{n=1}^{N} {p_n}^{w_n}$$

Here pn is the precision for n-grams. For N = 4 (up to 4-grams) with uniform weights.

$$Geometric\ Average\ Precision\ (4) = (p_1)^{\frac{1}{4}}.(p_2)^{\frac{1}{4}}.(p_3)^{\frac{1}{4}}.(p_4)^{\frac{1}{4}}$$
$$= (0.625)^{\frac{1}{4}}.(0.5714)^{\frac{1}{4}}.(0.5)^{\frac{1}{4}}.(0.4)^{\frac{1}{4}} = 0.5169$$

What is Brevity Penalty?

Imagine the scenario where the language model predicts only one word, such as “cancelled,” resulting in a clipped precision of 1. However, this can be misleading as it encourages the model to predict fewer words to achieve a high score.

To address this issue, a Brevity penalty is used, which penalizes machine translations that are too short compared to the reference sentence. Where, c is the predicted length i.e. number of words in the predicated sentence. “r” is the target length i.e. number of words in the target sentence.

Here, Brevity Penalty =1

So BLEU(4) = 0.5169*1 = 0.5169

How to Implement BLEU Score in Python?

There are various implementations of the BLEU score in Python under different libraries. We will be using evaluate library. Evaluate library simplifies the process of evaluating and comparing language model results.

Installation

```
!pip install evaluate import evaluate bleu = evaluate.load("bleu") predictions = ["They cancelled the match because it was raining "] references = ["They cancelled the match because of bad weather"] results = bleu.compute(predictions=predictions, references=references) print(results)
```

```
{'bleu': 0.5169731539571706,
 'precisions': [0.625, 0.5714285714285714, 0.5, 0.4],
 'brevity_penalty': 1.0,
 'length_ratio': 1.0,
 'translation_length': 8,
 'reference_length': 8}
```

BLEU Score Limitations

- It does not capture the semantic and syntactic similarity of the word. If the language model uses “called off” instead of “cancelled”, the bleu score considers it as an incorrect word.
- It doesn’t capture the significance of individual words within the text. For instance, prepositions, which typically carry less weight in meaning, are given equal importance by BLEU alongside nouns and verbs.
- It doesn’t preserve the order of words.
- It only considers exact word matches. For instance, “rain” and “raining” convey the same meaning, but BLEU Score treats them as errors due to the lack of exact match.
- It primarily relies on precision and doesn’t consider recall. Therefore, it doesn’t consider whether all words from the reference are included in the predicted text or not.

What is ROUGE score?

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) score comprises a set of metrics used for text summarization (commonly) and machine translation tasks evaluation. It was designed to evaluate the quality of machine-generated summaries by comparing them against the reference summaries. It measures the similarity between the machine-generated summary and the reference summaries by examining the overlapping n-grams. ROUGE metrics range from 0 to 1, where higher scores signify greater similarity between the automatically generated summary and the reference, whereas a score closer to zero suggests poor similarity between the candidate and the references.

Different Types of Metrics under ROUGE

ROUGE-N: Measures the overlap of n-grams between the system and reference summaries. For example, ROUGE-1 assesses the overlap of unigrams (individual words), whereas ROUGE-2 examines the overlap of bigrams (pairs of two consecutive words).

ROUGE-L: It relies on the length of the Longest Common Subsequence (LCS). It calculates the longest common subsequence (LCS) between the candidate text and the reference text. It doesn’t require consecutive matches but instead considers in-sequence matches, reflecting the word order at the sentence level.

ROUGE-Lsum: It divides the text into sentences using newlines and calculates the LCS for each pair of sentences. It then combines all LCS scores into a unified metric. This method is suitable for situations where both the candidate and reference summaries consist of multiple sentences.

ROUGE Score Calculation

ROUGE is essentially the F1 score derived from the precision and recall of n-grams. Precision (in the context of ROUGE) represents the proportion of n-grams in the prediction that also appear in the reference.

$$Precision = \frac{Number\ of\ overlapping\ n\_grams}{Total\ number\ of\ n\_grams\ in\ the\ candidate(predicated)\ summary}$$

Recall (in the context of ROUGE) is the proportion of reference n-grams that are also captured by the model-generated summary.

$$Recall = \frac{Number\ of\ overlapping\ n\_grams}{Total\ number\ of\ n\_grams\ in\ the\ reference\ summary}$$

$$ROUGE\ (F1\ Score) = 2\ \frac{Precision * Recall}{(Precision + Recall)}$$

Let's understand the ROUGE score calculation with the help of below example:

Candidate/Predicted Summary: He was extremely happy last night.

Reference/Target Summary: He was happy last night.

ROUGE1

Predicated 1-grams: ['He', 'was', 'extremely', 'happy', 'last', 'night']

Reference 1-grams: ['He', 'was', 'happy', 'last', 'night']

Overlapping 1-grams: ['He', 'was', 'happy', 'last', 'night']

Precision 1-gram = 5/6 = 0.83

Recall 1-gram = 6/6 = 1

ROUGE1 = (2*0.83*1) **/**
(0.83+1) = 0.9090

ROUGE2

Predicated 2-grams: ['He was', 'was extremely', 'extremely happy', 'happy last', 'last night']

Reference 2-grams: ['He was', 'was happy', 'happy last', 'last night']

Overlapping 2-grams: ['He was', 'happy last', 'last night']

Precision 2-gram = 3/5 = 0.6

Recall 2-gram = 3/4 = 0.75

ROUGE2 = (2*0.6*0.75) **/** (0.6+0.75) = 0.6666

How to Implement ROUGE Score in Python?

```
import evaluate rouge = evaluate.load('rouge') predictions = ["He was extremely happy last night"] references = ["He was happy last night"] results = rouge.compute(predictions=predictions,references=references) print(results)
```

```
{'rouge1': 0.9090909090909091,
 'rouge2': 0.6666666666666665,
 'rougeL': 0.9090909090909091,
 'rougeLsum': 0.9090909090909091}
```

ROUGE Score Limitations

- It does not capture the semantic similarity of the words.
- Its ability to detect order is limited, particularly when shorter n-grams are examined.
- It lacks a proper mechanism for penalizing specific prediction lengths, such as when the generated summary is overly brief or contains unnecessary details.

What is METEOR?

METEOR (Metric for Evaluation of Translation with Explicit Ordering) score is a metric used to assess the quality of generated text by evaluating the alignment between the generated text and the reference text. It is computed using the harmonic mean of precision and recall, with recall being weighted more than precision. METEOR also incorporates a chunk penalty (a measure of fragmentation), which is intended to directly assess how well-ordered the matched words in the machine translation are compared to the reference.

It is a generalized concept of unigram matching between the machine-generated translation and reference translations. Unigrams can be matched according to their original forms, stemmed forms, synonyms, and meanings. It ranges from 0 to 1, where a higher score indicates better alignment between the language model translated text and the reference text.

Key Features of METEOR

- It considers the order in which words appear as it penalizes the results having incorrect syntactical orders. BLEU score does not take word order into account.
- It incorporates synonyms, stems, and paraphrases, allowing it to recognize translations that use different words or phrases while still conveying the same meaning as the reference translation.
- Unlike the BLEU score, METEOR considers both the precision and recall (generally having more weight).
- Mathematically METEOR is defined as –

$$METEOR = (1 - chunk\ penalty) * Weighted\ F\_score$$

METEOR Score Calculation

Let's understand the BLEU score calculation using the following example:

Candidate/Predicted: The dog is hiding under the table.

Reference/Target: The dog is under the table.

Weighted F-score

Let's first compute the weighted F-score.

$$Weighted\ F\_score = \frac{Precision * Recall}{\alpha * Precision + (1 - \alpha) * Recall}$$

Where α parameter controls the relative weights of precision and recall, with a default value of 0.9.

Predicated 1-grams: ['The', 'dog', 'is', 'hiding', 'under', 'the', 'table']

Reference 1-grams: ['The', 'dog', 'is', 'under', 'the', 'table']

Overlapping 1-grams: ['The', 'dog', 'is', 'under', 'the', 'table']

Precision 1-gram = 6/7 = 0.8571

Recall 1-gram = 6/6 = 1

So weighted F-score = 0.9836

Chunk Penalty

To ensure the correct word order, a penalty function is incorporated that rewards the longest matches and penalizes the more fragmented matches. The penalty function is defined as –

$$Penalty = \gamma(\frac{c}{m})^{\beta}$$

Where β is the parameter that controls the shape of the penalty as a function of fragmentation. The default value is 3. Parameter determines the relative weight assigned to the fragmentation penalty. The default value is 0.5.

"c" is the number of longest matching chunks in the candidate, here {'the dog is', 'under the table'}. "m" is the number of unique unigrams in the candidate.

So Penalty = 0.0185

METEOR = (1 – Penalty) *
Weighted F-score = (1-0.0185)*0.9836 = 0.965

How to Implement METEOR Score in Python?

```
import evaluate meteor = evaluate.load('meteor') predictions = ["The dog is hiding under the table"] references = ["The dog is under the table"] results = meteor.compute(predictions=predictions,references=references) print(results)
```

```
{'meteor': 0.9653916211293262}
```

Conclusion

In this article, we discussed various types of quantitative metrics to evaluate the language model's output. We additionally delved into their computation, presenting it clearly and understandably through both mathematical concepts and code implementation.

Key Takeaways

- Assessing language models is essential to validate their output accuracy, efficiency, and reliability.
- BLEU and METEOR are primarily used for machine translation tasks in NLP and ROUGE for text summarization.
- The evaluate Python library contains built-in implementation for various quantitative metrics such as BLEU, ROUGE, METEOR, Perplexity, BERT score, etc.
- Capturing the contextual and semantic relationships is crucial when evaluating output, yet standard quantitative metrics often fall short in achieving this.

Frequently Asked Questions

Q1. What is the significance of the brevity penalty in the context of BLEU score?

A. Brevity Penalty addresses the potential issue of overly short translations produced by language models. Without the Brevity Penalty, a model could artificially inflate its score by predicting fewer words, which might not accurately reflect the quality of the translation. The penalty penalizes translations that are significantly shorter than the reference sentence.

Q2. What are the different types of metrics returned by the evaluate library while computing the ROUGE score?

A. The built-in implementation of the ROUGE score inside the evaluate library returns rouge1, rouge2, rougeL, and rougeLsum.

Q3. Out of the above three metrics which one makes use of recall?

A. ROUGE and METEOR make use of recall in their calculations, where METEOR assigns more weight to recall.

Article Url - https://www.analyticsvidhya.com/blog/2024/03/quantitative-metrics-simplified-for-language-model-evaluation/

Vikas Verma