Link

Mastering ROUGE Matrix: Your Guide to Large Language Model Evaluation for Summarization with Examples

#llm#nlp#machinelearning

Introduction

In this article, we will start discussing everything you might want to learn about the ROUGE Matrix. Let's begin by welcoming you to the world of Large Language Models (LLMs), where natural language processing has been revolutionized, becoming a cornerstone across various applications. However, with this innovation, the challenge of evaluating LLMs' or any text generation model performance and capabilities becomes visible. Traditional machine learning evaluation methods fall short due to the non-predictable nature of language-based outputs.

In the realm of Natural Language Processing (NLP), evaluating model performance is very important. But what does it mean when we say, "the model performed well on this task" or "the fine-tuned model showed substantial performance over the original model"? For this article, our focus turns to exploring the realm of ROUGE metrics in the context of summarization tasks. We'll dive into how these metrics shed light on the evaluation, reliability, and safety of Large Language Models (LLMs) when it comes to summarizing content. As we navigate through the process of evaluating summarization models, we'll uncover methodologies that enrich our understanding of the strengths and limitations of LLMs in creating summaries.

The Tricky Part of Model Evaluation in Language Stuff

In traditional machine learning, assessing a model's performance often involves comparing its predictions with known ground truth labels. Metrics like accuracy work well when predictions are deterministic. However, evaluating NLP models poses unique challenges due to the non-predictable nature of their outputs and the complexities of human language.

Imagine comparing two sentences: "John likes ice cream" and "John enjoys ice cream.” While these sentences convey similar meanings, determining their similarity isn't straightforward. Furthermore, understood differences in sentences like "John likes ice cream.” and "John doesn't like ice cream.” can lead to entirely different meanings. Evaluating such details requires an automated and structured approach, especially when dealing with large language models trained on vast amounts of text.

Meeting ROUGE and BLEU Measures

ROUGE and BLEU are tools that help us measure how well the computer understands language. ROUGE checks if the short summaries the computer makes are like what a person would write. It's like comparing a student's summary to the teacher's. BLEU is about checking if the computer's translations are good. It's like seeing how well the computer changes a sentence from one language to another and we will talk about this in the next article.

Understanding ROUGE Metrics

ROUGE is all about checking if the computer summaries make sense. Imagine your friend tells you a story, and you want to tell someone else the main parts of it. ROUGE helps us see if the computer did a good job at that. It looks at the important words the computer used and compares them to the words your friend used. ROUGE checks if they match. in other words it measures how well the machine-generated summaries match up with human-written reference summaries.

ROUGE works by emphasizing a recall-oriented assessment. This means it focuses on how much of the important content from the human-written summary is captured in the machine-generated summary. To achieve this, ROUGE examines various sets of words called n-grams, which are just word groups. For example,ROUGE-1, for instance, looks at individual words or unigrams, while ROUGE-2 considers pairs of words or bigrams, and so on. Additionally, ROUGE-L examines the longest common subsequence between the machine-generated and human reference summaries.

ROUGE-1: Capturing Unigrams

Let's dive into an example to understand ROUGE metrics better. Imagine a reference sentence created by a person: "The car is fast." Now, let's say the computer-generated summary reads: "The new red car is extremely incredibly fast." We want to assess how well the computer's output matches the reference. We use metrics like recall, precision, and F1 with ROUGE-1.

Reference sentence: "The car is fast."
Machine-Generated summary: "The new red car is extremely incredibly fast."

Recall:

measure of how many words from the machine-generated summary match words in the reference summary:

- Common terms in reference and machine-generated summaries are: "The" (1), "car" (1), "is" (1), "fast" (1) leading to `4` matched words.
- The total words in the reference summary are: "The" (1), "car" (1), "is" (1), "fast" (1), totaling `4` words.
- Therefore, the recall is `4/4 = 1`, because the machine-generated summary has correctly included all the words in the reference summary.

Precision

reflects the ratio of words in the machine-generated summary that match the words in the reference summary:

- With `4` matched words and the machine-generated summary including: "The" (1), "new" (1), "red" (1), "car" (1), "is" (1), "extremely" (1), "incredibly" (1), "fast" (1), summing up to `8 words`,
- The precision is `4/8 = 0.5`. Half of the words in the machine-generated summary are present in the reference summary.

F1 Score

This is a balance between recall and precision, giving us a single value to gauge overall performance. It's calculated using a formula that takes both recall and precision into account:

- Given our values, the F1 Score calculation is `2 * ((1 * 0.5) / (1 + 0.5))` approximately equal to `0.67`.

In this example:

- The recall is high because all the information in the reference text was captured in the generated text.
- However, the precision is lower because the generated text included extra information not in the reference text.
- This demonstrates that recall and precision can behave differently depending on the length and content of the generated summary.

In the upcoming sections of this article, we will concentrate solely on the recall aspect, as it is the central concern of ROUGE metrics. Yet, I wanted to inform you about all the common approaches to address this, as outlined in this section.

ROUGE-2: Capturing Bigrams

Suppose we have:

Reference sentence: "John is a talented musician."
Machine-Generated summary: "John is an accomplished artist."

- Breaking down each sentence into bigrams (pairs of words), we have:
    - **Bigrams in the reference summary:** "John is", "is a", "a talented", "talented musician"
    - **Bigrams in the machine-generated summary:** "John is", "is an", "an accomplished", "accomplished artist"

Overlap refers to the count of bigrams present in both the machine-generated and reference summaries. Upon inspecting the bigrams, we find that only the bigram "John is" appears in both summaries. As a result, the overlap count is `1`.

The ROUGE-2 Score is calculated by taking the number of overlapping bigrams and dividing it by the total number of bigrams in the reference summary.
Given `1` overlapping bigram and `4` total bigrams in the reference summary, the ROUGE-2 score computes to `1/4 = 0.25`.

Hence, in this instance, the ROUGE-2 score is `0.25`, indicating that 25% of the bigrams in the reference summary are present in the machine-generated summary.

ROUGE-L: Longest Common Subsequence

Let's examine the following sentences:

Reference sentence: "The quick brown fox jumps over the lazy dog."
Machine-Generated summary: "The quick dog jumps on the log."

The **Longest Common Subsequence (LCS)** method identifies the longest sequence of words that appear in the same order in both the reference and machine-generated summaries.

In this case, the LCS includes: "The quick" and "jumps the".

To calculate the **ROUGE-L Score**, we divide the length of the LCS by the total number of words in the reference summary.

Length of LCS: `4` words (for "The quick" and "jumps the")
Total words in reference: `9` words (for "The quick brown fox jumps over the lazy dog.")

Thus, the ROUGE-L score is computed as: `4/9 = 0.44`.

This indicates that around 44% of the reference summary is mirrored in the machine-generated summary when considering the longest common subsequence of words.

Keep in mind that in real-world contexts, the LCS typically emphasizes the longest sequence of consecutive words shared by both texts. The example provided here is a simplified and concatenated scenario intended for abstract illustration purposes.

ROUGE-Lsum: A Twist on ROUGE-L

The ROUGE-Lsum is related to the ROUGE-L metric but applies a slightly different calculation method. It applies the ROUGE-L calculation method at the sentence level and then aggregates all the results for the final score. This metric is seen as more suitable for tasks where sentence level extraction is valuable such as extractive summarization tasks.
In simpler terms, whereas ROUGE-L looks at the summary as a whole, ROUGE-Lsum considers sentence-level information, potentially providing more granularity in some use cases.

ROUGE-L ignores newlines and computes the LCS for the entire text. ROUGE-Lsum splits the text into sentences based on newlines and computes the LCS for each pair of sentences and take the average score for all sentences.

Example: calculating the ROUGE-L and ROUGE-Lsum

To truly grasp the beauty of these metrics, let's embark on an illustrative journey using an example:

Reference Summary (A): "John is a talented musician.`\n` He has band called (The Forest Rangers) Recently"
Generated Summary by System (B): "John is an accomplished artist.`\n` He is part of band called (The Forest Rangers)

ROUGE-L Calculation:

ROUGE-L looks at the whole story without caring about where lines break. It wants to find similar parts between the two versions.

In our example, it finds that LCS equals `9` common words: "John is a. He band called (The Forest Rangers)"

Thus, the ROUGE-L score for (A, B) would be `9/13 = 0.69` (approximated)

ROUGE-Lsum: Looking at Sentences Level.

ROUGE-Lsum first splits the summaries into sentences, then performs ROUGE-L calculations for each sentence individually:

Sentence 1: It compares the sentences "John is a talented musician." and "John is an accomplished artist." It finds that LCS equals `3` common words out of `5`, making the score `0.6`.

Sentence 2: Next, it compares "He has a band called (The Forest Rangers)." and "He is part of a band called (The Forest Rangers)." It finds that LCS equals `6` common words out of `8`, making the score `0.75`.

Adding these scores and finding the average gives us `(0.6 + 0.75) / 2`, which is **`0.675`**.

Key Differences: ROUGE-Lsum VS ROUGE-L

- ROUGE-L ignores newlines and computes the LCS for the entire text. This is appropriate when the candidate and reference summaries contain a single long sequence.
- ROUGE-L**sum* splits the text into sentences based on newlines and computes the LCS for each pair of sentences. It then takes the union of all LCS scores. This is appropriate when the candidate and reference summaries contain multiple sentences.
- ROUGE-L tends to give higher scores when the summaries contain similar content, regardless of sentence structure.
- ROUGE-Lsum penalizes differences in sentence structure more since it computes the LCS for each pair of sentences.

Calculating ROUGE Scores Using Python: A Step-by-Step Guide

Let's dive into the exciting world of calculating ROUGE scores using Python. Don't worry if coding isn't your comfort zone – we'll make it as smooth.

****Snippet 1: ROUGE Calculation with RougeScorer****

In this first snippet, we'll use a special tool called the **`rouge_score`** library. It's like a helper that understands ROUGE and does the math for us. Take a look:

```
# Import the rouge_scorer function from rouge_score
from rouge_score import rouge_scorer

# Initialize the scorer
scorer = rouge_scorer.RougeScorer(['rouge1', 'rouge2', 'rougeL', 'rougeLsum'])

# Compute the Rouge scores for reference and candidate.
scores = scorer.score('The quick brown fox jumps over the lazy dog',
                      'The quick brown dog jumps on the log.')

# Print the scores
print(scores)
```

In this snippet, you're using the **`RougeScorer`** class from the **`rouge_score`** library to create a scorer object for calculating ROUGE scores. The list of metrics you're calculating includes 'rouge1', 'rouge2', 'rougeL', and 'rougeLsum'. You then provide two sentences for comparison and print the resulting scores.

```
{
'rouge1': Score(precision=0.75, recall=0.67, fmeasure=0.71),
'rouge2': Score(precision=0.29, recall=0.25, fmeasure=0.27),
'rougeL': Score(precision=0.625, recall=0.56, fmeasure=0.59),
'rougeLsum': Score(precision=0.625, recall=0.56, fmeasure=0.59)
}
```

****Snippet 2: Aggregated ROUGE Scores with Evaluation Module****

Now, let's explore the second snippet. It's like inviting a group of friends to join the fun. We'll use an extra tool called the **`evaluate`** module, which is part of a library. It's like having a group activity to see how well sentences match up. Here's how it goes:

```
# Import the load function from the evaluate module
from evaluate import load

# Loading the 'rouge' metric from the library
rouge = load('rouge')

# Define your predictions and references
```

```
predictions = ["Your summary 1", "Your summary 2"]
references = ["Reference summary 1", "Reference summary 2"]

# Compute the scores
results = rouge.compute(predictions=predictions, references=references)

# Print the scores
print(results)
```

Here, you're loading the 'rouge' metric from the library using the `load` function. Then, you define a list of predictions and corresponding reference summaries. By calling the `compute` function on the `rouge` metric object, you calculate aggregated ROUGE scores for all predictions and references.

The resulting dictionary `results` contains the aggregated ROUGE scores for 'rouge1', 'rouge2', 'rougeL', and 'rougeLsum'.

```
{
'rouge1': 0.67,
'rouge2': 0.5,
'rougeL': 0.67,
'rougeLsum': 0.67
}
```

Also you can see a real world example for ROUGE Metrics in comparing a FLAN-T5 model on this notebook.

Overcoming Challenges: Enhancing ROUGE Metrics

While ROUGE metrics offer valuable insights into language evaluation, they come with their quirks. Think of it like this – sometimes ROUGE gives a thumbs up when it shouldn't. Imagine a computer repeating the same word over and over, like "fix, fix, fix, fix." ROUGE might support this, but that's not great writing, right? To figure this out, we have a trick we use clipping function "rule" that can be used to limit the counting of repeated words "Don't count the same word too much.". This keeps things fair and accurate, making sure our evaluation truly reflects the quality of the generated summary.

Now, let's get into the nitty-gritty (details). ROUGE metrics have been our trusty companions in grading text summaries and translating stuff. But just like any measuring tool, they have their own limitations. Let's look at some of these:

1. **Word Friends, but Not BFFs**: ROUGE loves comparing words, but it's not so great at telling if words are like cousins – different on the outside but similar on the inside. This can be a problem when words have different faces but mean similar things. Imagine "happy" and "joyful" – they're like word twins, but ROUGE doesn't always see that.
2. **Word Order? Not a Big Deal**: ROUGE checks if words like to be together, but it doesn't care much about where they stand. This can be tricky. Imagine shuffling the words in a sentence – ROUGE might still think it's all good even if the meaning changes a lot. A bit like mixing up the ingredients in a recipe!
3. **Friends Matter**: ROUGE relies on its friends, the reference texts. If these friends talk a certain way, ROUGE kind of wants others to talk like that too. It's like a classroom where everyone wants to be friends with the popular kid. This can make things a bit biased, don't you think?
4. **Short and Sweet**: Ever notice that shorter things sometimes get more attention? Same goes for ROUGE scores. Shorter summaries can get a gold star just for being short, even if they don't capture everything. We have a trick to stop this, but it's not a perfect fix.

But fear not! We're not just here to point out problems. We've got solutions too:

1. **Mixing Meanings**: Let's bring in some friends – semantic similarity metrics. These guys help ROUGE understand that words with different looks can still have the same meaning. It's like teaching ROUGE to see beyond appearances.
2. **Order Matters**: Enter ROUGE-L as we mentioned. It cares about word order, so mixing things up doesn't fool it.
3. **Being Fair to Everyone**: Still working on new ways for ROUGE to make friends with all sorts of reference texts, not just the popular ones. This way, everyone gets a chance to shine.
4. **Tackling Shortness**: Still cooking up ways to be fair to both long and short summaries. So, even if you're brief, you won't get extra points just for keeping it short.

And guess what? researchers also thinking about blending different ROUGE versions together to get a full picture. It's like using different lenses to see a beautiful landscape – each lens gives us a unique view, and when we put them all together, we get the full beauty.

So, while ROUGE metrics are awesome, we're making them even better by tweaking, combining, and inviting some new friends to the party. Together, we're creating a fairer, more accurate way to measure the awesomeness of our language models. Stay tuned for more exciting updates in the world of language evaluation!

It's also good to know that there are other types of ROUGE, like ROUGE-W, ROUGE-S, and ROUGE-SU, which are not used as widely. These variations have different ways of looking at summaries, like weighted common words and skip-bigrams. While not as famous, they're still part of the family find more in Reference 4.

*In Conclusion*

So there you have it! To sum it up, ROUGE metrics are like guiding stars that help us understand how well language models work. We've learned about ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-Lsum, which help us figure out if computer-generated summaries match human ones.

Although ROUGE has some challenges, smart people are finding ways to fix them. They're making ROUGE better by teaching it to understand similar words, care about word order, treat everyone fairly, and not favor short things.

As we go forward, ROUGE metrics keep showing us the way to better language models. They're helping us see the true abilities of these models and making sure they work even better. So, keep an eye out for more exciting things in the world of language evaluation!

Thanks for reading ❤

References

1. https://github.com/google-research/google-research/tree/master/rouge
2. https://github.com/huggingface/datasets/blob/main/metrics/rouge
3. https://github.com/huggingface/datasets/issues/617
4. https://aclanthology.org/W04-1013.pdf
5. https://huggingface.co/spaces/evaluate-metric/rouge
6. In accordance with SEO best practices, the canonical article can be found here.