MACHINE LEARNING

Skip-Bigrams In System

By bomber bot April 23, 2024

Automatically generating summaries of long text documents is a challenging natural language processing task. But how do we know if a machine-generated summary is actually any good? That's where ROUGE comes in.

ROUGE, which stands for Recall-Oriented Understudy for Gisting Evaluation, is a set of metrics for evaluating both automatic summarization and machine translation of texts. It was originally proposed by Chin-Yew Lin in a 2004 paper and has since become the standard for assessing summarization systems.

The key idea behind ROUGE is to compare an automatically produced summary to one or more human-written reference summaries and count the overlapping units such as word n-grams, word sequences, and word pairs. The intuition is that a high-quality machine summary should share many words and phrases with a human-written summary of the same source text.

Precision, Recall, And F-Measure

To understand how ROUGE works, we first need to define three important metrics: recall, precision, and F-measure.

Recall measures what fraction of the reference summary was captured or recovered by the system summary. We can calculate it as:

```
               # of overlapping words
recall = --------------------------
            # of words in reference
```

So let's say we have the following system and reference summaries:

```
System summary: The cat was found under the bed.
Reference summary: The cat was hidden under the bed.
```

The overlapping words are: "The", "cat", "was", "under", "the", "bed". So the recall is 6/7 = 0.857. In other words, the system summary recovered 85.7% of the words in the reference summary.

However, high recall alone doesn‘t mean the system summary is good. It could be capturing all the right words but also including a lot of irrelevant information. That‘s why we also need to measure precision.

Precision tells us what fraction of the words in the system summary were relevant or needed. We can calculate it as:

```
            # of overlapping words
precision = -------------------------
            # of words in system
```

In the example above, the precision is 6/6 = 1.0, meaning every word in the system summary was relevant. But if the system summary contained many extra words:

```
System summary: The furry cat was suspiciously found under the wooden bed.
```

Then the precision would only be 6/10 = 0.6. So while recall measures how much of the reference was recovered, precision measures how much of what was recovered is actually relevant.

Since there is often a trade-off between precision and recall, it‘s common to combine them into a single metric called F-measure, which is the harmonic mean of precision and recall:

```
              2 * precision * recall
F-measure = -----------------------
               precision + recall
```

A perfect F-measure of 1.0 requires both perfect precision and perfect recall. In practice, summarization systems aim to achieve a good balance of the two.

ROUGE-N: N-Grams

The original ROUGE metric, ROUGE-N, measures the overlap of n-grams between the system and reference summaries. An n-gram is simply a contiguous sequence of n items, which can be words, characters, or other units.

ROUGE-1 looks at unigram (one-word) overlap, ROUGE-2 looks at bigram overlap, ROUGE-3 looks at trigram overlap, and so on. The higher the n-gram order, the more strictly we are measuring the similarity between the system and reference.

To calculate ROUGE-N precision, recall, and F-measure, we simply use n-grams instead of individual words in the equations above. Let‘s calculate ROUGE-2 recall for the example from before:

```
System summary: The cat was found under the bed.
Reference summary: The cat was hidden under the bed.
```

The overlapping bigrams are: "The cat", "under the", "the bed". The total number of reference bigrams is 5. So the ROUGE-2 recall is 3/5 = 0.6. And the ROUGE-2 precision is 3/6 = 0.5.

Here is how we could implement ROUGE-N in Python:

```
import itertools
def rouge_n(system, reference, n):
sys_ngrams = list(itertools.ngrams(system.split(), n))
ref_ngrams = list(itertools.ngrams(reference.split(), n))


overlaps = set(sys_ngrams) & set(ref_ngrams)
recall = len(overlaps) / len(ref_ngrams)
precision = len(overlaps) / len(sys_ngrams)


if precision + recall == 0:
f_measure = 0
else:
f_measure = 2  precision  recall / (precision + recall)
```

```
    return {"recall": recall, "precision": precision, "f-measure": f_measure}


system = "The cat was found under the bed."
reference = "The cat was hidden under the bed."


print(rouge_n(system, reference, 1)) # ROUGE-1
print(rouge_n(system, reference, 2)) # ROUGE-2
```

This prints:

```
{'recall': 0.8571428571428571, 'precision': 1.0, 'f-measure':
0.9230769230769231}
{'recall': 0.6, 'precision': 0.5, 'f-measure': 0.5454545454545455}
```

As we increase n, the ROUGE-N scores get lower since there are fewer overlapping n-grams between the system and reference. In general, ROUGE-1 and ROUGE-2 are the most commonly reported metrics.

ROUGE-L: Longest Common Subsequence

One limitation of ROUGE-N is that it only considers consecutive matches. But sometimes a system summary may contain fragments of the reference text rearranged in a different order. ROUGE-L addresses this by identifying the Longest Common Subsequence (LCS) between the system and reference.

The LCS is the longest sequence of words that appear in both summaries, in the same order, allowing for gaps. For example:

```
System summary: The quick dog jumps over the lazy fox.
Reference summary: The quick brown fox jumps over the lazy dog.
```

The LCS is: "The quick fox jumps over the lazy". Compared to ROUGE-N, ROUGE-L is less sensitive to word order and better captures the fluency and coherence of the system summary.

To calculate ROUGE-L, we first find the length of the LCS between the system and reference summaries. Then we compute precision and recall as:

```
               LCS length
ROUGE-L recall = ------------------------
                Reference length
```

LCS length

```
ROUGE-L precision = -----------------------
```

System length

And the F-measure is calculated the same way as before. Here's a Python implementation:

```
def lcs(X, Y):
  m = len(X)
  n = len(Y)
L = [[None]*(n+1) for i in range(m+1)]


for i in range(m+1):
for j in range(n+1):
if i == 0 or j == 0:
L[i][j] = 0
```

```
elif X[i-1] == Y[j-1]:
L[i][j] = L[i-1][j-1]+1
else:
L[i][j] = max(L[i-1][j], L[i][j-1])



return L[m][n]



def rouge_l(system, reference):
sys_len = len(system.split())
ref_len = len(reference.split())
lcs_len = lcs(system.split(), reference.split())



recall = lcs_len / ref_len
precision = lcs_len / sys_len



if precision + recall == 0:
f_measure = 0
else:

f_measure = 2  precision  recall / (precision + recall)



return {"recall": recall, "precision": precision, "f-measure":
f_measure}
```

```
system = "The quick dog jumps over the lazy fox."

reference = "The quick brown fox jumps over the lazy dog."


print(rouge_l(system, reference))
```

This prints:

```
{'recall': 0.7777777777777778, 'precision': 0.875, 'f-measure':
0.823529411764706}
```

So while the system and reference summaries share many of the same words, their word order differs, resulting in a moderately high ROUGE-L score.

ROUGE-S: Skip-Bigrams

Another variant is ROUGE-S, which measures the overlap of skip-bigrams between the system and reference summaries. Skip-bigrams are pairs of words in a sentence order, allowing for gaps in between.

For example, the skip-bigrams in the phrase "the cat in the hat" are:

```
("the", "cat"), ("the", "in"), ("the", "the"), ("the", "hat"), ("cat",
"in"), ("cat", "the"), ("cat", "hat"), ("in", "the"), ("in", "hat"),
("the", "hat")
```

To limit the number of skip-bigrams considered, we can specify a maximum skip distance, ROUGE-S4 considers pairs of words with up to 4 words in between.

Like with ROUGE-N, precision and recall are computed by counting the number of overlapping skip-bigrams between the system and reference summaries:

```
                  # overlapping skip-bigrams
ROUGE-S recall = -----------------------------
                  # skip-bigrams in reference
# overlapping skip-bigrams
```

```
ROUGE-S precision = ---------------------------
```

Here is how to implement ROUGE-S in Python:

```
from itertools import combinations
def skipbigrams(sequence, n):
return set(combinations(sequence, 2))


def rouge_s(system, reference, n=2):

sys_skipbigrams = skipbigrams(system.split(), n)
ref_skipbigrams = skipbigrams(reference.split(), n)


overlaps = sys_skipbigrams & ref_skipbigrams
recall = len(overlaps) / len(ref_skipbigrams)
precision = len(overlaps) / len(sys_skipbigrams)
```

```
if precision + recall == 0:
f_measure = 0
else:
f_measure = 2  *precision*  recall / (precision + recall)


return {"recall": recall, "precision": precision, "f-measure":
f_measure}


system = "The quick dog jumps over the lazy fox."
reference = "The quick brown fox jumps over the lazy dog."


print(rouge_s(system, reference))
```

This prints:

```
{‘recall‘: 0.35, ‘precision‘: 0.4166666666666667, ‘f-measure‘:
0.38095238095238093}
```

The skip-bigram overlap rewards summaries with similar long-distance word pairs as the reference, even if the short-range word order is different. However, ROUGE-S is not as widely used as ROUGE-N and ROUGE-L.

Best Practices

When evaluating a summarization system with ROUGE, it‘s recommended to:

- Use multiple reference summaries per document to capture diversity in human summaries and reduce bias. The final ROUGE score can be the average across references.
- Report ROUGE-1, ROUGE-2, and ROUGE-L scores as a standard since they capture different aspects of overlap.
-

Apply stemming and remove stop words, punctuation, and case distinctions before counting matches. This puts more emphasis on content words. Most ROUGE implementations do this by default.

- Include confidence intervals to show the variance in ROUGE scores across test examples. This gives a sense of the stability of the summarizer‘s performance.

- Use the same ROUGE parameters (e.g. n-gram order, stemming) when comparing different summarization systems to ensure a fair comparison. The ROUGE toolkit makes it easy to standardize settings.

While ROUGE is the most popular summarization metric, it‘s not perfect. Some argue that it focuses too much on lexical overlap and not enough on semantic similarity, coherence, or readability. There are other evaluation metrics like METEOR, CIDEr, and BERTScore that try to address these limitations.

Additionally, ROUGE may not correlate well with human judgments, especially for abstractive summaries that deviate from the original wording. Therefore, it‘s important to also conduct manual evaluation with human raters to assess the fluency, informativeness, and overall quality of system summaries.

State Of The Art

Over the past decade, neural network models have achieved remarkable improvements on ROUGE benchmarks. As of 2021, some of the top performing summarization systems include:

- PEGASUS (Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence): a Transformer-based model pre-trained on a novel self-supervised objective that masks out important sentences and generates them as the target. It achieved ROUGE-1/2/L scores of 44.17/21.47/41.11 on the CNN/DailyMail dataset.
- BART (Bidirectional and Auto-Regressive Transformers): a denoising autoencoder that combines bidirectional encoding with

autoregressive generation. It achieved ROUGE-1/2/L scores of 44.16/21.28/40.90 on CNN/DailyMail.

- T5 (Text-to-Text Transfer Transformer): an extremely large pre-trained language model that frames various NLP tasks as text-to-text problems. It achieved ROUGE-1/2/L scores of 43.52/21.55/40.69 on CNN/DailyMail.

These models demonstrate the power of large-scale self-supervised pre-training and fine-tuning for abstractive summarization. However, they require substantial computing resources to train and run.

For more efficient models, distilled versions like DistilBART and DistilPEGASUS can achieve comparable performance with faster inference speed and smaller model size. Extractive summarization methods based on sentence similarity and centrality can also be effective for certain use cases.

Conclusion

ROUGE is an essential tool for automatically evaluating summarization systems. By comparing the n-gram, LCS, and skip-bigram overlaps between machine-generated and human-written summaries, ROUGE provides a quick and reproducible way to measure progress on this challenging NLP task.

When used properly with multiple references, standard parameters, and human evaluation, ROUGE helps guide research toward more informative, fluent, and human-like text summarization. As models become increasingly powerful, this will enable a wide range of applications like personalized news digests, meeting minutes, legal contract analysis, and scientific literature reviews.

Equipped with this knowledge of ROUGE, you are now ready to assess summarization systems and contribute to the exciting field of NLP! Get out there and start ROUGEing up some summaries.

#Machine Learning

← PREVIOUS

What is REST? Rest API Definition for Beginners

NEXT →

What is R Squared? R2 Value Meaning and Definition

Similar Posts

How To Develop An End-To-End Machine Learning Project And Deploy It To Heroku With Flask

By bomber bot April 19, 2024

Why Most Startups Should Outsource Their Machine Learning Work

By bomber bot April 24, 2024

About FAQ Privacy Policy Terms of use