Foundations of NLP Explained — Bleu Score and WER Metrics

A Gentle Guide to two essential metrics (Bleu Score and Word Error Rate) for NLP models, in Plain English

Most NLP applications such as machine translation, chatbots, text summarization, and language models generate some text as their output. In addition applications like image captioning or automatic speech recognition (ie. Speech-to-Text) output text, even though they may not be considered pure NLP applications.

How good is the predicted output?

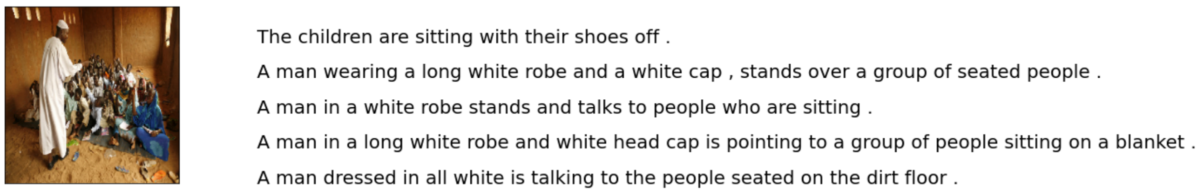
The common problem when training these applications is how do we decide how 'good' that output is?

With applications like, say, image classification the predicted class can be compared unambiguously with the target class to decide whether the output is correct or not. However, the problem is much trickier with applications where the output is a sentence.

In such a case we don't always have one universally correct answer — we could have many correct answers. When translating a sentence, for instance, two different people may come up with two slightly different answers, both of which are completely correct.

eg. “*The ball is blue*” and “*The ball has a blue color*”.

The problem is even harder with applications like image captioning or text summarization, where the range of acceptable answers is even larger.



The children are sitting with their shoes off .
A man wearing a long white robe and a white cap , stands over a group of seated people .
A man in a white robe stands and talks to people who are sitting .
A man in a long white robe and white head cap is pointing to a group of people sitting on a blanket .
A man dressed in all white is talking to the people seated on the dirt floor .

The same image can have many valid captions (Image by Author)

In order to evaluate the performance of our model, we need a quantitative metric to measure the quality of its predictions.

In this article, I will cover two major NLP metrics which you are likely to use fairly frequently with your NLP models.

Also, if you are interested in NLP, I have a few articles that explore other fascinating topics in this space.

1. Automatic Speech Recognition *(Speech-to-Text algorithm and architecture, using CTC Loss and Decoding for aligning sequences.)*
2. Beam Search (*how Beam Search enhances predictions in NLP models*)

NLP Metric

Over the years a number of different NLP metrics have been developed to tackle this problem. One of the most popular is called the Bleu Score.

It is far from perfect, and it has many drawbacks. But it is simple to compute and understand and has several compelling benefits. Even though it has many alternatives, it continues to be one of the most frequently used metrics.

It is based on the idea that the closer the predicted sentence is to the human-generated target sentence, the better it is.

Bleu Scores are between 0 and 1. A score of 0.6 or 0.7 is considered the best you can achieve. Even two humans would likely come up with different sentence variants for a problem, and would rarely achieve a perfect match. For this reason, a score closer to 1 is unrealistic in practice and should raise a flag that your model is overfitting.

Before we get into how Bleu Score is calculated, let’s understand two concepts first viz. N-grams and Precision.

N-gram

An ‘n-gram’ is actually a widely used concept from regular text processing and is not specific to NLP or Bleu Score. It is just a fancy way of describing “a set of ’n’ consecutive words in a sentence”.

For instance, in the sentence “The ball is blue”, we could have n-grams such as:

- 1-gram (unigram): “The”, “ball”, “is”, “blue”
- 2-gram (bigram): “The ball”, “ball is”, “is blue”
- 3-gram (trigram): “The ball is”, “ball is blue”
- 4-gram: “The ball is blue”

Note that the words in an n-gram are taken in order, so “blue is The ball” is not a valid 4-gram.

Precision

This metric measures the number of words in the Predicted Sentence that also occur in the Target Sentence.

Let’s say, that we have:

- **Target Sentence**: He eats an apple
- **Predicted Sentence**: He ate an apple

We would normally compute the Precision using the formula:

Precision = Number of correct predicted words / Number of total predicted words

Precision = 3 / 4

But using Precision like this is not good enough. There are two cases that we still need to handle.

Repetition

The first issue is that this formula allows us to cheat. We could predict a sentence:

- **Target Sentence**: He eats an apple
- **Predicted Sentence**: He He He

and get a perfect Precision = 3 / 3 = 1

Multiple Target Sentences

Secondly, as we've already discussed, there are many correct ways to express the same sentence. In many NLP models, we might be given multiple acceptable target sentences that capture these different variations.

We account for these two scenarios using a modified Precision formula which we'll call "Clipped Precision".

Clipped Precision

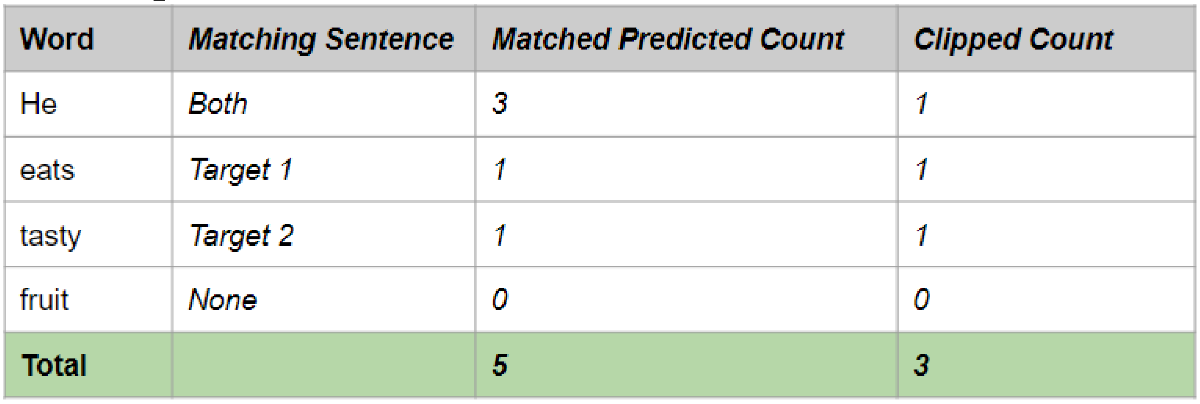
Let's go through an example to understand how it works.

Let’s say, that we have the following sentences:

- **Target Sentence 1**: He eats a sweet apple
- **Target Sentence 2**: He is eating a tasty apple
- **Predicted Sentence**: He He He eats tasty fruit

We now do two things differently:

- We compare each word from the predicted sentence with all of the target sentences. If the word matches any target sentence, it is considered to be correct.
- We limit the count for each correct word to the maximum number of times that that word occurs in the Target Sentence. This helps to avoid the Repetition problem. This will become clearer below.



Word	*Matching Sentence*	*Matched Predicted Count*	*Clipped Count*
He	*Both*	*3*	*1*
eats	*Target 1*	*1*	*1*
tasty	*Target 2*	*1*	*1*
fruit	*None*	*0*	*0*
Total		**5**	**3**

Clipped Precision (Image by Author)

For instance, the word “He” occurs only once in each Target Sentence. Therefore, even though “He” occurs thrice in the Predicted Sentence, we ‘clip’ the count to one, as that is the maximum count in any Target Sentence.

Clipped Precision = Clipped number of correct predicted words / Number of total predicted words

Clipped Precision = 3 / 6

NB: For the rest of this article, we will just use "Precision" to mean "Clipped Precision".

We are now ready to go ahead and calculate the Bleu Score.

How is Bleu Score calculated?

Let's say we have an NLP model that produces a predicted sentence as below. For simplicity, we will take just one Target Sentence, but as in the example above, the procedure for multiple Target Sentences is very similar.

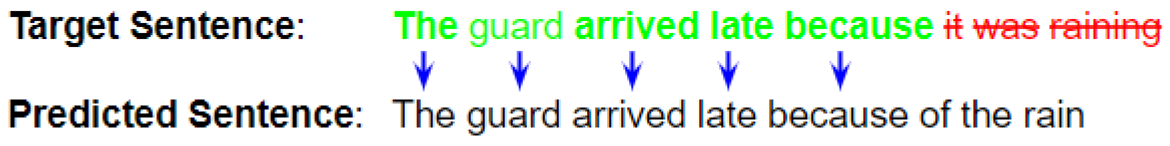
- **Target Sentence**: The guard arrived late because it was raining
- **Predicted Sentence**: The guard arrived late because of the rain

The first step is to compute Precision scores for 1-grams through 4-grams.

Precision 1-gram

We use the Clipped Precision method that we just discussed.

Precision 1-gram = Number of correct predicted 1-grams / Number of total predicted 1-grams


Target Sentence: The guard arrived late because ~~it was raining~~
Predicted Sentence: The guard arrived late because of the rain

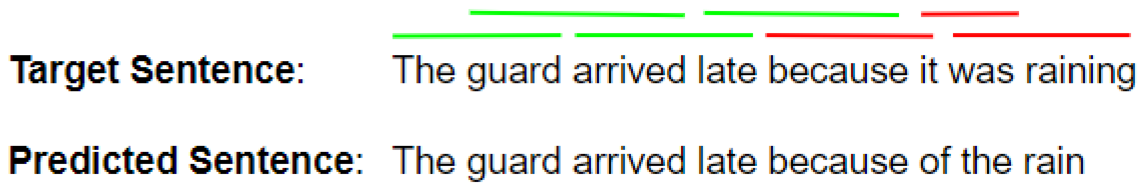
Precision(Image by Author)

So, Precision 1-gram (*p1*) = 5 / 8

Precision 2-gram

Precision 2-gram = Number of correct predicted 2-grams / Number of total predicted 2-grams

Let's look at all the 2-grams in our predicted sentence:

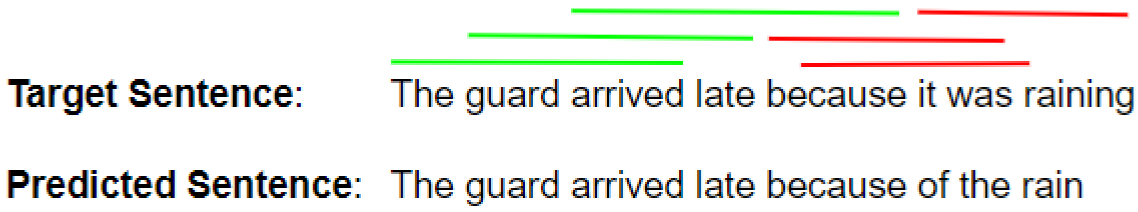

Target Sentence: The guard arrived late because it was raining
Predicted Sentence: The guard arrived late because of the rain

Precision 2-gram (Image by Author)

So, Precision 2-gram (*p2*) = 4 / 7

Precision 3-gram

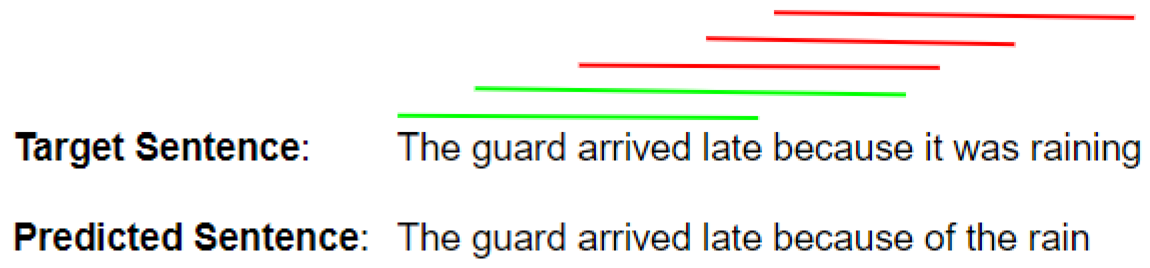
Similarly, Precision 3-gram (*p3*) = 3 / 6


Target Sentence: The guard arrived late because it was raining
Predicted Sentence: The guard arrived late because of the rain

Precision 3-gram (Image by Author)

Precision 4-gram

And, Precision 4-gram (*p4*) = 2 / 5


Target Sentence: The guard arrived late because it was raining

Predicted Sentence: The guard arrived late because of the rain

Precision 4-gram(Image by Author)

Geometric Average Precision Scores

Next, we combine these Precision Scores using the formula below. This can be computed for different values of N and using different weight values. Typically, we use $N = 4$ and uniform weights $w_n = N / 4$

$$\begin{aligned} Geometric\ Average\ Precision\ (N) &= exp\left(\sum_{n=1}^{N} w_n \ log\ p_n\right) \\ &= \prod_{n=1}^{N} {p_n}^{w_n} \\ &= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \end{aligned}$$

Precision Scores (Image by Author)

Brevity Penalty

The third step is to compute a 'Brevity Penalty'.

If you notice how Precision is calculated, we could have output a predicted sentence consisting of a single word like "The' or "late". For this, the 1-gram Precision would have been 1/1 = 1, indicating a perfect score. This is obviously misleading because it encourages the model to output fewer words and get a high score.

To offset this, the Brevity Penalty penalizes sentences that are too short.

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

Brevity Penalty(Image by Author)

- c is *predicted length = number of words in the predicted sentence* and
- r is *target length = number of words in the target sentence*

This ensures that the Brevity Penalty cannot be larger than 1, even if the predicted sentence is much longer than the target. And, if you predict very few words, this value will be small.

In this example, c = 8 and r = 8, which means Brevity Penalty = 1

Bleu Score

Finally, to calculate the Bleu Score, we multiply the Brevity Penalty with the Geometric Average of the Precision Scores.

$$Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N)$$

Bleu Score (Image by Author)

Bleu Score can be computed for different values of N. Typically, we use N = 4.

- BLEU-1 uses the unigram Precision score
- BLEU-2 uses the geometric average of unigram and bigram precision

- BLEU-3 uses the geometric average of unigram, bigram, and trigram precision
- and so on.

If you look at different resources on the Internet, you might also encounter a slightly different way to write the Bleu Score formula, which is mathematically equivalent.

$$
\begin{aligned}
log\ Bleu &= min(1-\frac{r}{c},0)+\sum_{n=1}^{4}\frac{log\ p_n}{4} \\
&= min(1-\frac{r}{c},0)+\frac{log\ p_1+log\ p_2+log\ p_3+log\ p_4}{4}
\end{aligned}
$$

Bleu Score formula (Image by Author)

Implementing Bleu Score in Python

In practice, you will rarely have to implement the Bleu Score algorithm on your own. The nltk library, which is a very useful library for NLP functionality, provides an implementation of Bleu Score.

```python
from nltk.translate.bleu_score import corpus_bleu

references = [[['my', 'first', 'correct', 'sentence'], ['my', 'second', 'valid', 'sentence']]]
candidates = [['my', 'sentence']]
score = corpus_bleu(references, candidates)
```

Now that we know how Bleu Score works, there are a few more points that we should note.

Bleu Score is computed on a corpus, not individual sentences

Although we used examples of matching single sentences, Bleu Score is calculated by considering the text of the entire predicted corpus as a whole.

Therefore you cannot compute Bleu Score separately on each sentence in the corpus, and then average those scores in some way.

Strengths of Bleu Score

The reason that Bleu Score is so popular is that it has several strengths:

- It is quick to calculate and easy to understand.
- It corresponds with the way a human would evaluate the same text.
- Importantly, it is language-independent making it straightforward to apply to your NLP models.
- It can be used when you have more than one ground truth sentence.
- It is used very widely, which makes it easier to compare your results with other work.

Weaknesses of Bleu Score

In spite of its popularity, Bleu Score has been criticized for its weaknesses:

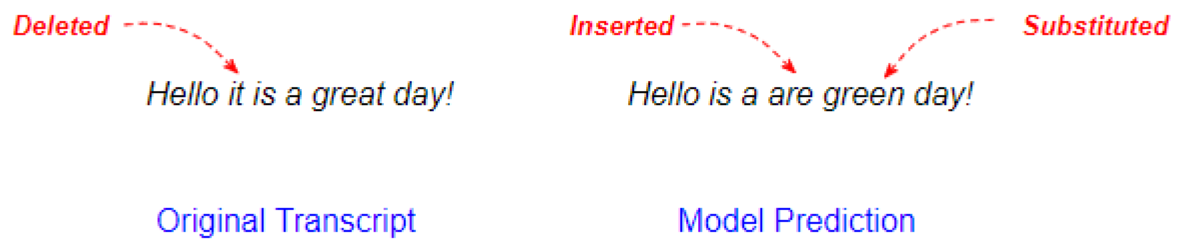
- It does not consider the meaning of words. It is perfectly acceptable to a human to use a different word with the same meaning eg. Use “watchman” instead of “guard”. But Bleu Score considers that an incorrect word.
- It looks only for exact word matches. Sometimes a variant of the same word can be used eg. “rain” and “raining”, but Bleu Score counts that as an error.
- It ignores the importance of words. With Bleu Score an incorrect word like “to” or “an” that is less relevant to the sentence is penalized just as heavily as a word that contributes significantly to the meaning of the sentence.
- It does not consider the order of words eg. The sentence “The guard arrived late because of the rain” and “The rain arrived late because of the guard” would get the same (unigram) Bleu Score even though the latter is quite different.

Speech-to-Text applications use Word Error Rate, not Bleu Score

Although Automatic Speech Recognition models also output text, the target sentence is unambiguous and usually not subject to interpretation. In this case, Bleu Score is not the ideal metric.

The metric that is typically used for these applications is Word Error Rate (WER), or its sibling, Character Error Rate (CER). It compares the predicted output and the target transcript, word by word (or character by character) to figure out the number of differences between them.

A difference could be a word that is present in the transcript but missing from the prediction (counted as a Deletion), a word that is not in the transcript but has been added into the prediction (an Insertion), or a word that is altered between the prediction and the transcript (a Substitution).



Count the Insertions, Deletions, and Substitutions between the Transcript and the Prediction (Image by Author)

The metric formula is fairly straightforward. It is the percent of differences relative to the total number of words.

$$\text{Word Error Rate} = \frac{\text{Inserted} + \text{Deleted} + \text{Substituted}}{\text{Total words in transcript}} = \frac{1 + 1 + 1}{6} = 0.5$$

Word Error Rate computation (Image by Author)

The WER calculation is based on the Levenstein distance, which measures the differences between two words.

Although WER is the most widely used metric for Speech Recognition, it has some drawbacks:

- It does not distinguish between words that are important to the meaning of the sentence and those that are not as relevant.
- When comparing words, it does not consider whether two words are different in just a single character or are completely different.