A Gentle Introduction to Calculating the BLEU Score for Text in Python

by **Jason Brownlee** on November 20, 2017 in **Deep Learning for Natural Language Processing**

Last Updated on December 19, 2019

BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate translation of text to one or more reference translations.

Although developed for translation, it can be used to evaluate text generated for a suite of natural language processing tasks.

In this tutorial, you will discover the BLEU score for evaluating and scoring candidate text using the NLTK library in Python.

After completing this tutorial, you will know:

- A gentle introduction to the BLEU score and an intuition for what is being calculated.
- How you can calculate BLEU scores in Python using the NLTK library for sentences and documents.
- How you can use a suite of small examples to develop an intuition for how differences between a candidate and reference text impact the final BLEU score.

Tutorial Overview

This tutorial is divided into 4 parts; they are:

1. Bilingual Evaluation Understudy Score

2. Calculate BLEU Scores
3. Cumulative and Individual BLEU Scores
4. Worked Examples

Bilingual Evaluation Understudy Score

The Bilingual Evaluation Understudy Score, or BLEU for short, is a metric for evaluating a generated sentence to a reference sentence.

A perfect match results in a score of 1.0, whereas a perfect mismatch results in a score of 0.0.

The score was developed for evaluating the predictions made by automatic machine translation systems. It is not perfect, but does offer 5 compelling benefits:

- It is quick and inexpensive to calculate.
- It is easy to understand.
- It is language independent.
- It correlates highly with human evaluation.
- It has been widely adopted.

The BLEU score was proposed by Kishore Papineni, et al. in their 2002 paper "BLEU: a Method for Automatic Evaluation of Machine Translation".

In addition to translation, we can use the BLEU score for other language generation problems with deep learning methods such as:

- Language generation.
- Image caption generation.

- Text summarization.
- Speech recognition.

And much more.

Calculate BLEU Scores

The Python Natural Language Toolkit library, or NLTK, provides an implementation of the BLEU score that you can use to evaluate your generated text against a reference.

Sentence BLEU Score

NLTK provides the sentence_bleu() function for evaluating a candidate sentence against one or more reference sentences.

The reference sentences must be provided as a list of sentences where each reference is a list of tokens. The candidate sentence is provided as a list of tokens. For example:

```
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'a', 'test'], ['this', 'is' 'test
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate)
print(score)
```

Running this example prints a perfect score as the candidate matches one of the references exactly.

```
1.0
```

Corpus BLEU Score

NLTK also provides a function called corpus_bleu() for calculating the BLEU score for multiple sentences such as a paragraph or a

document.

The references must be specified as a list of documents where each document is a list of references and each alternative reference is a list of tokens, e.g. a list of lists of lists of tokens. The candidate documents must be specified as a list where each document is a list of tokens, e.g. a list of lists of tokens.

This is a little confusing; here is an example of two references for one document.

```
# two references for one document
from nltk.translate.bleu_score import corpus_bleu
references = [[['this', 'is', 'a', 'test'], ['this', 'is' 'te
candidates = [['this', 'is', 'a', 'test']]
score = corpus_bleu(references, candidates)
print(score)
```

Running the example prints a perfect score as before.

```
1.0
```

Cumulative and Individual BLEU Scores

The BLEU score calculations in NLTK allow you to specify the weighting of different n-grams in the calculation of the BLEU score.

This gives you the flexibility to calculate different types of BLEU score, such as individual and cumulative n-gram scores.

Let's take a look.

Individual N-Gram Scores

An individual N-gram score is the evaluation of just matching grams of a specific order, such as single words (1-gram) or word pairs (2-gram or bigram).

The weights are specified as a tuple where each index refers to the gram order. To calculate the BLEU score only for 1-gram matches, you can specify a weight of 1 for 1-gram and 0 for 2, 3 and 4 (1, 0, 0, 0). For example:

```
# 1-gram individual BLEU
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'small', 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate, weights=(1, 0, 0,
print(score)
```

Running this example prints a score of 0.5.

```
0.75
```

We can repeat this example for individual n-grams from 1 to 4 as follows:

```
# n-gram individual BLEU
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'a', 'test']]
candidate = ['this', 'is', 'a', 'test']
print('Individual 1-gram: %f' % sentence_bleu(reference, cand
print('Individual 2-gram: %f' % sentence_bleu(reference, cand
print('Individual 3-gram: %f' % sentence_bleu(reference, cand
print('Individual 4-gram: %f' % sentence_bleu(reference, cand
```

Running the example gives the following results.

```
Individual 1-gram: 1.000000
Individual 2-gram: 1.000000
Individual 3-gram: 1.000000
Individual 4-gram: 1.000000
```

Although we can calculate the individual BLEU scores, this is not how the method was intended to be used and the scores do not carry a lot of meaning, or seem that interpretable.

Cumulative N-Gram Scores

Cumulative scores refer to the calculation of individual n-gram scores at all orders from 1 to n and weighting them by calculating the weighted geometric mean.

By default, the *sentence_bleu()* and *corpus_bleu()* scores calculate the cumulative 4-gram BLEU score, also called BLEU-4.

The weights for the BLEU-4 are 1/4 (25%) or 0.25 for each of the 1-gram, 2-gram, 3-gram and 4-gram scores. For example:

```
# 4-gram cumulative BLEU
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'small', 'test']]
candidate = ['this', 'is', 'a', 'test']
score = sentence_bleu(reference, candidate, weights=(0.25, 0.
print(score)
```

Running this example prints the following score:

```
1.0547686614863434e-154
```

The cumulative and individual 1-gram BLEU use the same weights, e.g. (1, 0, 0, 0). The 2-gram weights assign a 50% to each of 1-gram and 2-gram and the 3-gram weights are 33% for each of the 1, 2 and 3-gram scores.

Let's make this concrete by calculating the cumulative scores for BLEU-1, BLEU-2, BLEU-3 and BLEU-4:

```
# cumulative BLEU scores
from nltk.translate.bleu_score import sentence_bleu
reference = [['this', 'is', 'small', 'test']]
candidate = ['this', 'is', 'a', 'test']
print('Cumulative 1-gram: %f' % sentence_bleu(reference, cand
print('Cumulative 2-gram: %f' % sentence_bleu(reference, cand
print('Cumulative 3-gram: %f' % sentence_bleu(reference, cand
print('Cumulative 4-gram: %f' % sentence_bleu(reference, cand
```

Running the example prints the following scores. They are quite different and more expressive than the

They are quite different and more expressive than the standalone individual n-gram scores.

```
Cumulative 1-gram: 0.750000
Cumulative 2-gram: 0.500000
Cumulative 3-gram: 0.000000
Cumulative 4-gram: 0.000000
```

It is common to report the cumulative BLEU-1 to BLEU-4 scores when describing the skill of a text generation system.

Worked Examples

In this section, we try to develop further intuition for the BLEU score with some examples.

We work at the sentence level with a single reference sentence of the following:

> *the quick brown fox jumped over the lazy dog*

First, let's look at a perfect score.

```
# prefect match
```

```
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'quick', 'brown', 'fox', 'jumped', 'over'
score = sentence_bleu(reference, candidate)
print(score)
```

Running the example prints a perfect match.

```
1.0
```

Next, let's change one word, '*quick*' to '*fast*'.

```
# one word different
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'fast', 'brown', 'fox', 'jumped', 'over',
score = sentence_bleu(reference, candidate)
print(score)
```

This result is a slight drop in score.

```
0.7506238537503395
```

Try changing two words, both '*quick*' to '*fast*' and '*lazy*' to '*sleepy*'.

```
# two words different
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'fast', 'brown', 'fox', 'jumped', 'over',
score = sentence_bleu(reference, candidate)
print(score)
```

Running the example, we can see a linear drop in skill.

```
0.4854917717073234
```

What about if all words are different in the candidate?

```
# all words different
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

```
score = sentence_bleu(reference, candidate)
print(score)
```

We get the worse possible score.

```
0.0
```

Now, let's try a candidate that has fewer words than the reference (e.g. drop the last two words), but the words are all correct.

```
# shorter candidate
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'quick', 'brown', 'fox', 'jumped', 'over'
score = sentence_bleu(reference, candidate)
print(score)
```

The score is much like the score when two words were wrong above.

```
0.7514772930752859
```

How about if we make the candidate two words longer than the reference?

```
# longer candidate
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'quick', 'brown', 'fox', 'jumped', 'over'
score = sentence_bleu(reference, candidate)
print(score)
```

Again, we can see that our intuition holds and the score is something like "*two words wrong*".

```
0.7860753021519787
```

Finally, let's compare a candidate that is way too short: only two words in length.

```
# very short
```

```
from nltk.translate.bleu_score import sentence_bleu
reference = [['the', 'quick', 'brown', 'fox', 'jumped', 'over
candidate = ['the', 'quick']
score = sentence_bleu(reference, candidate)
print(score)
```

Running this example first prints a warning message indicating that the 3-gram and above part of the evaluation (up to 4-gram) cannot be performed. This is fair given we only have 2-grams to work with in the candidate.

```
UserWarning:
Corpus/Sentence contains 0 counts of 3-gram overlaps.
BLEU scores might be undesirable; use SmoothingFunction().
  warnings.warn(_msg)
```

Next, we can a score that is very low indeed.

```
0.0301973834223185
```

I encourage you to continue to play with examples.

The math is pretty simple and I would also encourage you to read the paper and explore calculating the sentence-level score yourself in a spreadsheet.

Further Reading

This section provides more resources on the topic if you are looking go deeper.

- BLEU on Wikipedia
- BLEU: a Method for Automatic Evaluation of Machine Translation, 2002.
- Source code for nltk.translate.bleu_score

- nltk.translate package API Documentation

Summary

In this tutorial, you discovered the BLEU score for evaluating and scoring candidate text to reference text in machine translation and other language generation tasks.

Specifically, you learned:

- A gentle introduction to the BLEU score and an intuition for what is being calculated.
- How you can calculate BLEU scores in Python using the NLTK library for sentences and documents.
- How to can use a suite of small examples to develop an intuition for how differences between a candidate and reference text impact the final BLEU score.

Do you have any questions?
Ask your questions in the comments below and I will do my best to answer.