BERTScore: It's Better in Sesame Street.

Introduction

The paper proposes a new metric for evaluating generated text from a natural language model, checking the fluency of a given output using contextual text embeddings.

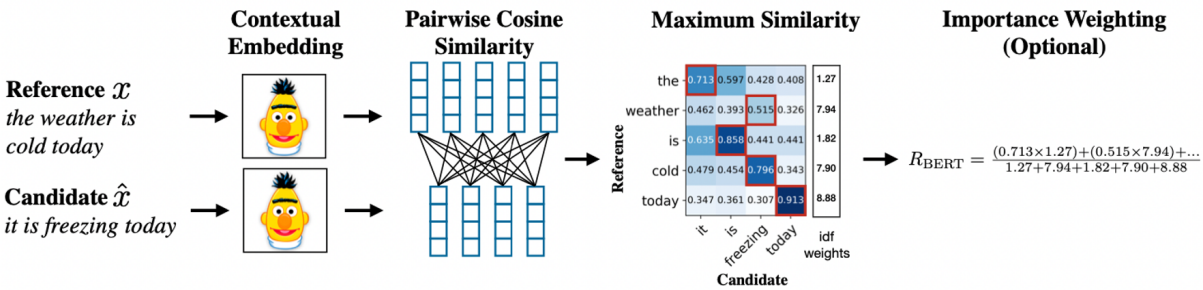
Previous metrics to evaluate text generation have majorly involved the idea of ngram-matching, counting the occurence of ngrams in both the reference and the target sentence. Some of the metrics include BLEU, METEOR, NIST, etc.

BLEU, the most commonly used ngram-matching based metric, allows each ngram in the reference to only match to the target once, and averages various scores obtained from different window sizes n.

Some approaches also opt to use individual word embeddings and shallow parses, such as MEANT 2.0, etc. These embeddings are commonly static, that is, once trained, a single word can only have one representation.

In contrast, the BERTScore metric uses BERT embeddings, which are context-bound; each word has a representation that is dependent on the surrounding environment (context) of the word. This helps account for cases of polysemy (where one word can take more than one meaning) - and overall helps in capturing the meaning of the token more effectively.

BERTScore



A sentence pair is individually decomposed into a contextual embedding for each token; using these embeddings, we obtain the maximum cosine similarity by the means of greedy matching. Finally these similarity scores are averaged, either directly by the number of tokens in the reference sentence (R_{BERT}) or the candidate sentence (P_{BERT}) - or by using term-wise IDF scores.

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \quad , \quad P_{\text{BERT}} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j \quad , \quad F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}} \quad .$$

As this approach is based on the cosine similarity of two vectors, it may take values varying from -1 to 1. To remedy this, and provide a more readable score, the BERTScore is rescaled with a baseline b.

This baseline is computed by creating 1 million random candidate-reference pairs (two random sentences) from a Common Crawl Monolingual Dataset for each language. As these two sentences are randomly picked, it is assumed that each pair will have low lexical and semantic overlap. b is then computed by averaging the BERTScore computed on these sentence pairs.

The rescaled value can be expressed as:

$\hat{R}_{BERT} = \frac{R_{BERT} - b}{1 - b}$

The same approach is used to rescale values of P_{BERT} and F_{BERT}.

Experimental Setup

The metric was used to analyse outputs in various different tasks in Natural Language Processing and Generation, to check whether the metric proposed is task-independent, and also to compare against previous metrics used in these domains.

Machine Translation

The WMT18 metric evaluation dataset was used, which contains predictions of 149 translation systems across 14 language pairs; gold references, and two types of human judgment scores, at a system level and at a segment level.

To evaluate metrics, we refer to the Pearson correlations between the score and the prediction.

Image Captioning

Referring to the COCO 2015 Captioning Challenge, the metric was tested to determine the correlation between the score and (M1) - the percentage of captions that were evaluated better/equal to human captions and (M2) - the percentage of captions that were indistinguishable from human captions.

Robustness Analysis

Used BERTScore and other common metrics on pairs from the Quora Question Pair Corpus; and also adversial paraphrases from the PAWS dataset.

Results

Machine Translation:

- BERTScore ranks high in system-level correlations.
- BERTScore shows better performance over BLEU, RUSE, etc.

Image Captioning:

- BERTScore performs better than the baseline measures.

Robustness:

Performs better than other metrics, even in case of adversial examples like in the PAWS dataset.

Strengths of the Paper

- ## Simple

  The approach suggested by the paper is straightforward and easy to implement/refine and develop as is required.

- ## Detailed Analysis of Benchmarks

  While also ensuring that the overall analysis is task-independent, it also provides comparitive analysis with many other metrics used, across various tasks in the domain of NLP.

- ## Better Understanding of Text

  BERT's contextual embeddings are a significant improvement from the ngram-matching based approaches which are largely devoid of meaning and based directly on phrase/clause matching.

Weaknesses of the Paper

- ## Computationally Intensive

  By virtue of the greedy matching approach, the process of computing the BERTScore for a given pair of sentences is computationally intensive as it requires $m * n$ operations where m is the number of tokens in the reference and n is the number of tokens in the target sentences.

- ## No Pragmatic Considerations for Text Generation

  This point may have differing levels of importance, but is mentioned nonetheless due to the paper's evaluation for text generation in the context of machine translation.

  A text that is generated in Hindi from a sentence in English: `He is wise like an owl` when translated into Hindi, may output `वह उल्लू की तरह बुद्धिमान है` - however, this does not account for the semantic gaps between English and Hindi, where `owl` is associated with `wisdom` and may be used to describe an old, wise ascetic; on the other hand, the Hindi counterpart `उल्लू` is more commonly associated to simpletons and `foolishness`.

A simple greedy matching approach will not help in such translation, and is also expected to struggle in case of Multi-Word Expressions (MWEs) and idiomatic expressions.

- ## IDF Weighting

  The effectiveness of using/not using IDF weighting while computing BERTScores is not regular; the paper however does not provide sufficient details on when it may be useful, and when it may be acceptable to omit this step.

Improvements to the Paper

- ## Determining when and why IDF Weighting is useful

  More analysis of IDF Weighting across different tasks and text domains may help in determining how IDF Weighting affects certain use cases and not in others.

- ## Optimizations in Computation Time

  Could try to make the score computation more effective, perhaps using some memoization; or using smaller models that compute similarity in lesser time.

- ## Usage of Linguistically Motivated Features

  More metrics for semantic similarity may be introduced to ensure the quality of generated text.