NLP Error Metrics:

BLEU:

BLEU: BiLingual Evaluation Understudy

NLP evaluation metric used in Machine Translation tasks

Suitable for measuring corpus level similarity

n

-gram comparison between words in candidate sentence and reference sentences

Range: 0 (no match) to 1 (exact match)

Example

```
hyp = str('she read the book because she was interested in world
history').split()

ref_a = str('she read the book because she was interested in world
history').split()

ref_b = str('she was interested in world history because she read the
book').split()
```

```
score_ref_a = bleu.sentence_bleu([ref_a], hyp)

print("Hyp and ref_a are the same: {}".format(score_ref_a))

score_ref_b = bleu.sentence_bleu([ref_b], hyp)

print("Hyp and ref_b are different: {}".format(score_ref_b))

score_ref_ab = bleu.sentence_bleu([ref_a, ref_b], hyp)

print("Hyp vs multiple refs: {}".format(score_ref_ab))



Hyp and ref_a are the same: 1.0

Hyp and ref_b are different: 0.7400828044922853
```

```
Hyp vs multiple refs: 1.0



hyp = str('she read the book because she was interested in world
history').split()

ref_a = str('she was interested in world history because she read the
book').split()

hyp_b = str('the book she read was about modern
civilizations.').split()

ref_b = str('the book she read was about modern
civilizations.').split()



score_a = bleu.sentence_bleu([ref_a], hyp)

score_b = bleu.sentence_bleu([ref_b], hyp_b)

score_ab = bleu.sentence_bleu([ref_a], hyp_b)

score_ba = bleu.sentence_bleu([ref_b], hyp)

score_ref_a = bleu.corpus_bleu([[ref_a], [ref_b]], [hyp, hyp_b])

average = (score_a+score_b)/2

corpus = math.pow((11+8)/19 * (9+7)/(17) * (6+6)/(9+6) * (4+5)/(8+5),
1/4)

print("Sent: {}, {}, {}, {} - Corpus {}, {}, {}".format(score_a,
score_b, score_ab, score_ba, score_ref_a, average, corpus))



Sent: 0.7400828044922853, 1.0, 0.44677880536150705, 0.5491004867761125
- Corpus 0.8496988908521796, 0.8700414022461427, 0.8496988908521795
```

BLEU Score: Bilingual Evaluation Understudy

BLEU is a precision-focused metric that measures the n-gram overlap between the generated text and the reference text. The score also considers a brevity penalty where a penalty is applied when the machine-generated text is too short compared to reference text. It is a metric that is generally used for machine translation performance. The score ranges from 0 to 1, with higher scores indicating greater similarity between the generated text and the reference text.

$$\text{BLEU} = \min\left(1, \frac{\text{output-length}}{\text{reference-length}}\right)\left(\prod_{i=1}^{4} \text{precision}_i\right)^{\frac{1}{4}}$$

2. Topic Modeling :

Metrics :

What is Topic Modeling?

- *Topic modeling is an **unsupervised** learning method, whose objective is to extract the underlying semantic patterns among a collection of texts. These underlying semantic structures are commonly referred to as **topics** of the corpus.*
- *In particular, topic modeling first extracts features from the words in the documents and use mathematical structures and frameworks like matrix factorization and SVD (Singular Value Decomposition) to identify clusters of words that share greater semantic coherence.*
- *These clusters of words form the notions of topics.*
- *Meanwhile, the mathematical framework will also determine the distribution of these **topics** for each document.*

- *In short, an intuitive understanding of Topic Modeling:*
    - *Each **document** consists of several **topics** (a distribution of different topics).*
    - *Each topic is connected to particular groups of **words** (a distribution of different words).*

Latent Dirichlet Allocation

Intuition of LDA

- *Latent Dirichlet [diɾiˈkleː] Allocation learns the relationships between **words**, **topics**, and **documents** by assuming documents are generated by a particular probabilistic model.*
- *A topic in LDA is a multinomial distribution over the words in the vocabulary of the corpus. (That is, given a topic, it's more likely to see specific sets of words).*

- *What LDA gives us is:*
    - *Which words are more likely to be connected to specific topics? (Topic by Word Matrix)*
    - *Which topics are more likely to be connected to specific documents? (Document by Topic Matrix)*

- *To interpret a topic in LDA, we examine the ranked list of the most probable (top N) words in that topic.*
- *Common words in the corpus often appear near the top of the words for each topic, which makes it hard to differentiate the meanings of these topics sometimes.*
- *When inspecting the word rankings for each topic, we can utilize two types of information provided by LDA:*
    - *The frequencies of the words under each topic*
    - *The exclusivity of the words to the topic (i.e., the degree to which the word appears in that particular topic to the exclusion of others).*

Building LDA Model

- *We should use `CountVectorizer` when fitting LDA instead of `TfidfVectorizer` because LDA is based on term count and document count.*

- *Fitting LDA with `TfidfVectorizer` will result in rare words being dis-proportionally sampled.*
- *As a result, they will have greater impact and influence on the final topic distribution.*

Model Performance Metrics

- *We can diagnose the model performance using **perplexity** and **log-likelihood**.*
    - *The higher the log-likelihood, the better.*
    - ***The lower the perplexity, the better.***

*In essense, since **perplexity** is equivalent to the inverse of the geometric mean, a lower perplexity implies data is more likely. As such, as the number of topics increase, the **perplexity** of the model should decrease*

*3. **Question and Answers:***

Metrics :

Metrics for QA

There are two dominant metrics used by many question answering datasets, including SQuAD: exact match (EM) and F1 score. These scores are computed on individual question+answer pairs. When multiple correct answers are possible for a given question, the maximum score over all possible correct answers is computed. Overall EM and F1 scores are computed for a model by averaging over the individual example scores.

Exact Match

This metric is as simple as it sounds. For each question+answer pair, if the characters of the model's prediction exactly match the characters of (one of) the True Answer(s), EM = 1, otherwise EM = 0. This is a strict all-or-nothing metric; being off by a single character results in a score of 0. When assessing against a negative example, if the model predicts any text at all, it automatically receives a 0 for that example.

F1

F1 score is a common metric for classification problems, and widely used in QA. It is appropriate when we care equally about precision and recall. In this case, it's computed over the individual words in the prediction against those in the True Answer. The number of shared words between the prediction and the truth is the basis of the F1 score: precision is the ratio of the number of shared words to the total number of words in the prediction, and

recall is the ratio of the number of shared words to the total number of words in the ground truth.

```
def compute_exact_match(prediction, truth):
    return int(normalize_text(prediction) == normalize_text(truth))


def compute_f1(prediction, truth):
    pred_tokens = normalize_text(prediction).split()
    truth_tokens = normalize_text(truth).split()

    # if either the prediction or the truth is no-answer then f1 = 1 if
they agree, 0 otherwise
    if len(pred_tokens) == 0 or len(truth_tokens) == 0:
        return int(pred_tokens == truth_tokens)

    common_tokens = set(pred_tokens) & set(truth_tokens)

    # if there are no common tokens then f1 = 0
    if len(common_tokens) == 0:
        return 0

    prec = len(common_tokens) / len(pred_tokens)
    rec = len(common_tokens) / len(truth_tokens)

    return 2 * (prec * rec) / (prec + rec)
```

4. ***Sentence Similarity***

- *Levenshtein distance*
- *Jaccard distance*
- *Euclidean distance*
- *Cosine similarity*
- *Similarity with [sklearn, gensim, spacy]*
- *Similarity with word embeddings (Word2vec similarity, Word movers distance)*
- *Similarity of Probability Distributions (Cross entropy, KL Divergence, Hellinger distance)*
- *Nearest-neighbors search*
- *Clustering*

"we often want to determine ***similarity between pairs of documents****, or the* ***similarity between a specific document*** *and a set of other documents (such as a user query vs. indexed documents).*

Use cases:

Broad:

- *information retrieval*
- *document clustering*
- *word-sense disambiguation*
- *automatic essay scoring*
- *short answer grading*
- *machine translation*
- *Recommendation Engines*
- *Search Engines (Query - Result matching)*

Specific:

- *Grant similarity*
- *Complaint similarity*
- *Duplicate questions*
- *Question Answering (e.g. give the same style of question to a customer support agent)*

5. Text summarization *:*

The next metric being discussed in this series of posts about metrics used in Natural Language Processing is the Recall-Oriented Understudy for Gisting Evaluation(ROUGE)

ROUGE is an evaluation metric used to assess the quality of NLP tasks such as text summarization and machine translation.

True Positives are the matching n-grams between References and Candidates. The False Negatives can be thought of as the n-grams that are in the actual sentences(references) but not in the candidate. The False Positives are then the n-grams that are present in the Candidate but not in the References. Check out this blog post Recall, Precision, F1-Score if you need a refresher on concepts such as True Positives, False Positives, Recall and Precision etc.,

Hence, the recall and precision can be obtained as shown below:

$$Recall = \frac{\text{Number of matching n-grams}}{\text{Number of n-grams in the Reference}}$$

$$Precision = \frac{\text{Number of matching n-grams}}{\text{Number of n-grams in the Candidate}}$$

Recall and Precision can be complementary sometimes and hence it is necessary to combine both of them to achieve a balance. For this reason, the ROUGE is calculated as F1-score in Python. The value ranges from 0 to 1 and the closer the value is to 1, the better the quality of the prediction

Advantages

1. *Easy to calculate and understand and is usually used to evaluate text summarization systems.*
2. *ROUGE and its variants have worked well for summarization of single documents and short texts.*

ROUGE Score: Recall Oriented Understudy for Gisting Evaluation Score

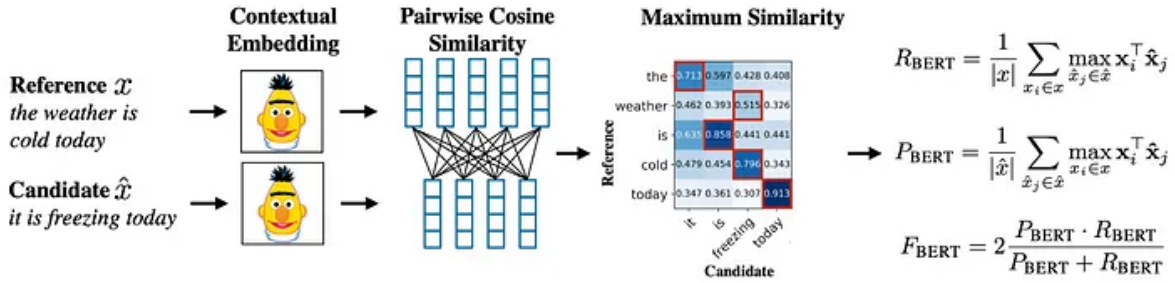
ROUGE is a metric that measures the overlap between the generated text and the reference text in terms of recall. Rouge comes in three types: rouge-n, the most prevalent form that detects n-gram overlap; rouge-l, which identifies the Longest Common Subsequence and rouge-s, which concentrates on skip grams. n-rouge is the most frequently used type with the following formula:

$$\frac{\text{number of n-grams found in model and reference}}{\text{number of n-grams in reference}}$$

BERT Score

One main disadvantage of using metrics such as BLEU or ROUGE is the fact that the performance of text generation models are dependent on exact matches. Exact matches might be important for use-cases like machine translation, however for generative AI models that try to create meaningful and similar texts to corpus data, exact matches might not be very accurate.

Introducing **BERTScore**



Source: Bertscore: Evaluating text generation with bert

Code for Bertscore is available at https://github.com/Tiiiger/bert_score

Hence, instead of exact matches, BERTScore is focused on the similarity between reference and generated text by using contextual embeddings. The main idea behind contextual embeddings is to understand the meaning behind the reference and candidate text respectively and then compare those meanings.