Semantic Similarity Detection

A BTP Report By

Hrishabh Pandey (**S20180010064**),
Ayush Gairola (**S20180010020**),
Rakesh Muchimari (**S20180010109**)

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

19 May 2020

1st Semester Report

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Semantic Similarity Detection**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2021 to May 2021 under the supervision of Dr. Amit Praseed, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date

Hrishabh Pandey

_ _

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date

(Dr. Amit Praseed)

Content

Introduction

What is semantic similarity?

Semantic Textual Similarity (STS) is defined as the measure of "**semantic equivalence**" between two blocks of text, phrases, sentences, or documents. Semantic similarity methods usually give a ranking or percentage of similarity between texts.

The main objective **Semantic Similarity** is to measure the distance between the semantic meanings of a pair of words, phrases, sentences, or documents.

For example, the word "car is moving" is more similar to "bus is moving" than it is to "cat is moving".

Why is it important?

Today we have an abundance of text data, but utilizing this resource is tricky as Computing sentence similarity is not a trivial task, due to the variability of natural language expressions. If done correctly, this metric could be used to collect, compare and classify large corpus of raw data, which could be utilized efficiently.

About the Project

For our BTP project, we took inspiration from the below listed implementations which are currently in use.

1. Medical : when a new case comes to a practitioner, he/she can look for similar cases in the past and get the most closely resembling case and make better decisions.
2. Law : When a new case comes, law firms look over their database for similar cases in the past and then strategies from those case studies. (Currently used by some firms)

These examples are a reality but are still kept as an internal tool by organizations, as it brings them a competitive edge.

We wish to present this technology in the hands of the general consumer with this platform where individuals and organizations will be able to collect and classify text data, which will accelerate their processes.

Comparison Metric

Cosine Similarity

Cosine similarity is a measure of **similarity** between two non-zero vectors of an inner product space. It is **defined** to be equal to the **cosine** of the angle between them, which is also the same as the inner product of the same vectors normalized to both have length 1.

Cos(x, y) = x . y / ||x|| * ||y||

- **x . y** = product (dot) of the vectors ‘x’ and ‘y’.
- **||x||** and **||y||** = length of the two vectors ‘x’ and ‘y’.
- **||x|| * ||y||** = cross product of the two vectors ‘x’ and ‘y’.

Advantages :

- The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.
- When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

Data Set Utilized

Sick DataSet

Marelli et al.compiled the SICK dataset for sentence level semantic similarity/relatedness in 2014 composed of 10,000 sentence pairs obtained from the ImageFlickr 8 and MSR-Video descriptions dataset. The sentence pairs were derived from image descriptions by various annotators. 750 random sentence pairs from the two datasets were selected,followed by three steps to obtain the final SICK dataset: sentence normalisation, sentence expansion and sentence pairing.

STS DataSet

In order to encourage research in the field of semantic similarity, semantic textual similarity tasks called SemEval have been conducted from 2012. The organizers of the SemEval tasks collected sentences from a wide variety of sources and compiled them to form a benchmark data set against which the performance of the models submitted by the participants in the task was measured

Methods Used

Word Count Semantic Similarity

Utilizing the Jaccard similarity index (sometimes called the Jaccard similarity coefficient) compares members for two sets to see which members are shared and which are distinct. It’s a measure of similarity for the two sets of data, with a range from 0% to 100%. The higher the percentage, the more similar the two populations.

The main idea here is to count the number of words that are common to both the sentences, divided by the total number of unique words.

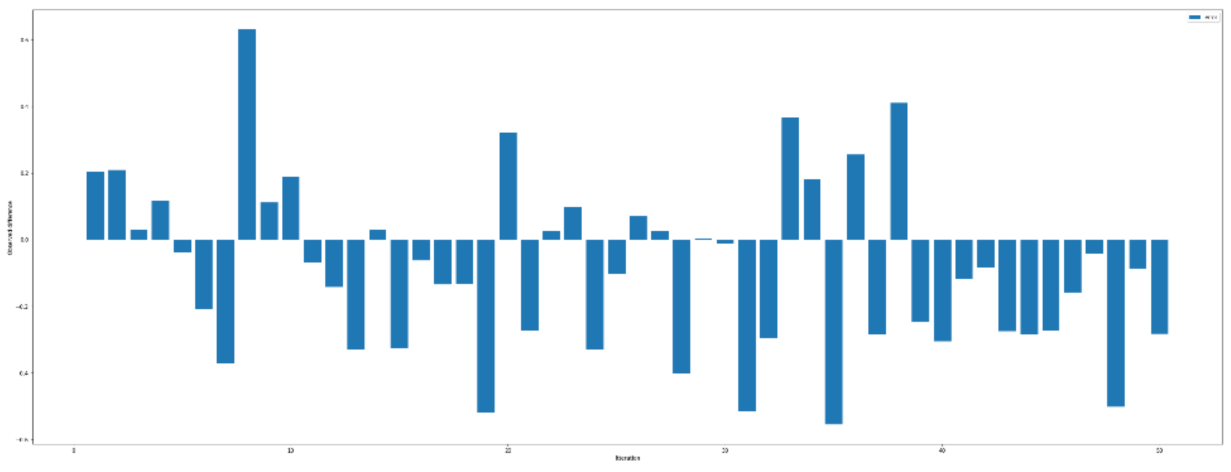
$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

A: Sentence 1 , B: Sentence 2
| A ∩ B | : Number of common words in A and B
| A ∪ B | : Number of unique words in A and B

Observations

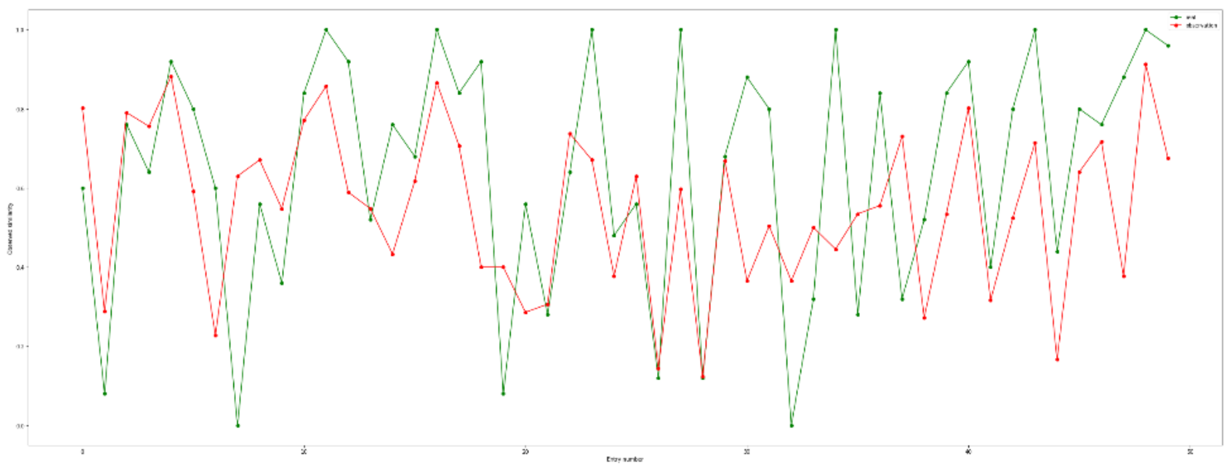
After utilizing the above metric, we observe that there was an “Average Difference between Expected and real Comparison Metrics” of -0.09 and “Variance of this difference Comparison Metrics” of 0.0652.
The reason for these good accuracy and low variances is the size of input. The test set contains sentences of length <= 10 words which makes this approach somewhat appealing.

Experimental Results	
Average Difference between Expected and real Comparison Metrics	-0.0901593368115178
Variance of this difference Comparison Metrics	0.06523802218468647



X axis : Sentences
Y axis : Difference between observed and real semantic similarity.

This graph shows the difference between read and calculate semantic similarity. For most of the query the difference is not much but in some cases with longer query the differences are too much leading to very high differences.



X axis : Sentences, **Red**: calculated Semantic Similarity, **Green**: Real Semantic Similarity.
Y axis :Plot of observed and real semantic similarity.

This plot gives a comparison of Real Semantic Similarity (Green) and Calculated Semantic Similarity (RED). Here it is observable that for the most part this metric works pretty fine for short queries but for long queries or sentences with synonyms, this metric fails.

Tf-Idf Semantic Similarity

TF-IDF stands for "**Term Frequency — Inverse Document Frequency**". This is a technique to quantify a word in documents, we generally compute a weight to each word which signifies the importance of the word in the document and corpus. By vectorizing the documents we can further perform multiple tasks such as finding the relevant documents, ranking, clustering and so on.

Term Frequency : This measures the frequency of a word in a document.

tf(t,d) = count of t in d / number of words in d

Document Frequency : DF is the number of documents in which the word is present.

df(t) = occurrence of t in documents

Inverse Document Frequency : IDF is the inverse of the document frequency which measures the informativeness of term t. When we calculate IDF, it will be very small for the most occurring words such as stop words . This finally gives what we want, a relative weightage. For large datasets (link corpus size 10,000) the idf term will explode. to dampen the effect we take of the IDF.

IDF(t) = log(N/(df + 1)) (df+1 to avoid division by zero error)

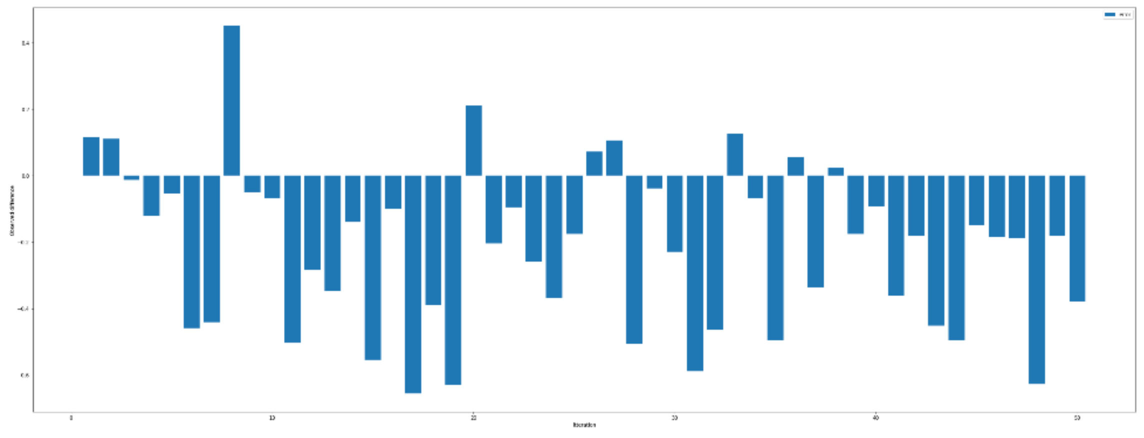
By taking a multiplicative value of TF and IDF, we get the TF-IDF score

TF-IDF(t, d) = tf(t, d) * log(N/(df + 1))

Observations

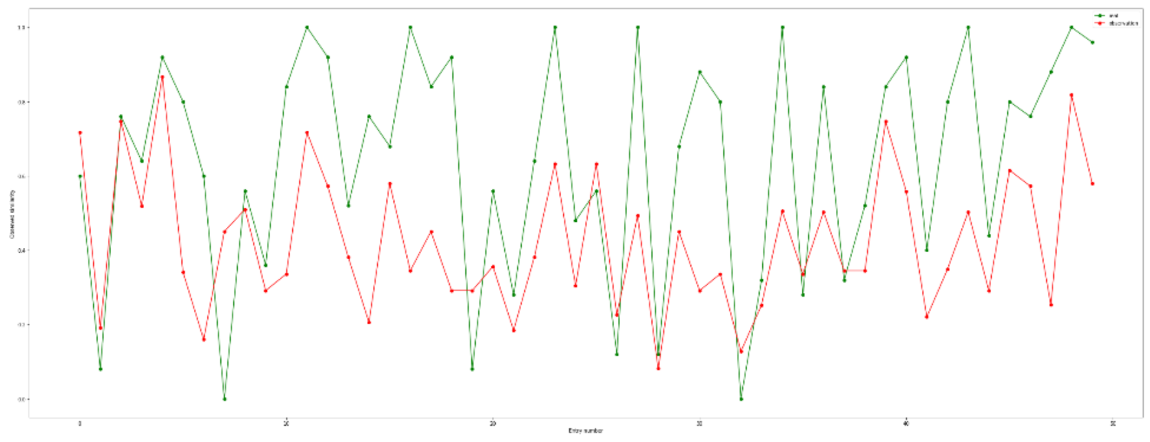
After utilizing the above metric, we observe that there was an "Average Difference between Expected and real Comparison Metrics" of -0.18109 and "Variance of this difference Comparison Metrics" of 0.0508. The reason for these good accuracy and low variances is the size of input. The test set contains sentences of length <= 10 words which makes this approach somewhat appealing.

Experimental Results	
Average Difference between Expected and real Comparison Metrics	-0.18109047743741563
Variance of this difference Comparison Metrics	0.050807551174712036



X axis : Sentences
Y axis : Difference between observed and real semantic similarity.

This graph shows the difference between read and calculate semantic similarity. For most of the query there is a negative difference, with low variance which indicates that the information captured is more aligned and now even though there is a difference in metric, the measures are aligned toward a particular side.



X axis : Sentences, **Red**: calculated Semantic Similarity, **Green**: Real Semantic Similarity.
Y axis :Plot of observed and real semantic similarity.

This plot gives a comparison of Real Semantic Similarity (Green) and Calculated Semantic Similarity (RED). Here it is observable that for the most part this metric works pretty fine for short queries but for long queries or sentences with synonyms, this metric fails.

Word Embedders Similarity

Creating sentence representation using word embeddings in vector form and comparing different sentences with cosine similarity. Utilizing the word embedder, we create a more robust vectorized representation of the sentence, which is then used for comparison using soft cosine similarity.

Word embeddings are models that convert text to vector representations obtained by training a neural network on a large corpus. It has been widely used in text classification using semantic similarity. For **example**, ‘apples’ and ‘oranges’ might be regarded as more similar than ‘apples’ and ‘Jupiter’.

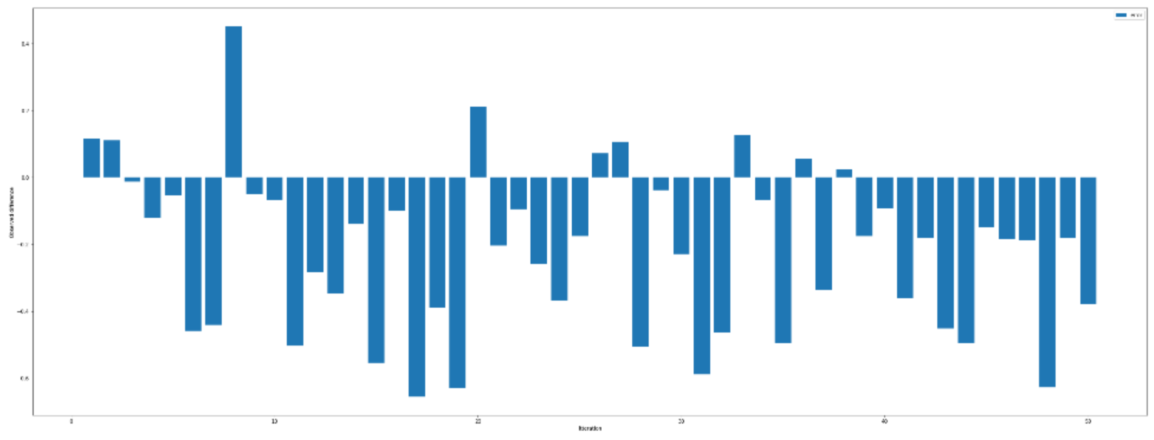
Utilizing Word embedders, we will be able to create vectors which would encapsulate meaning in a digitally computable way, allowing for easy comparison of sequences.

Observations

After utilizing the above metric, we observe that there was an “Average Difference between Expected and real Comparison Metrics” of 0.25 and “Variance of this difference Comparison Metrics” of 0.0604.
Here we Observe that we have got a higher error but a decent variance in errors, indicating that the word embedder is able to compare similarity between similar contextual words allowing for a better measurement. The reason for high error is insufficient training of word embedders and diversity of text used to train the word embedder.

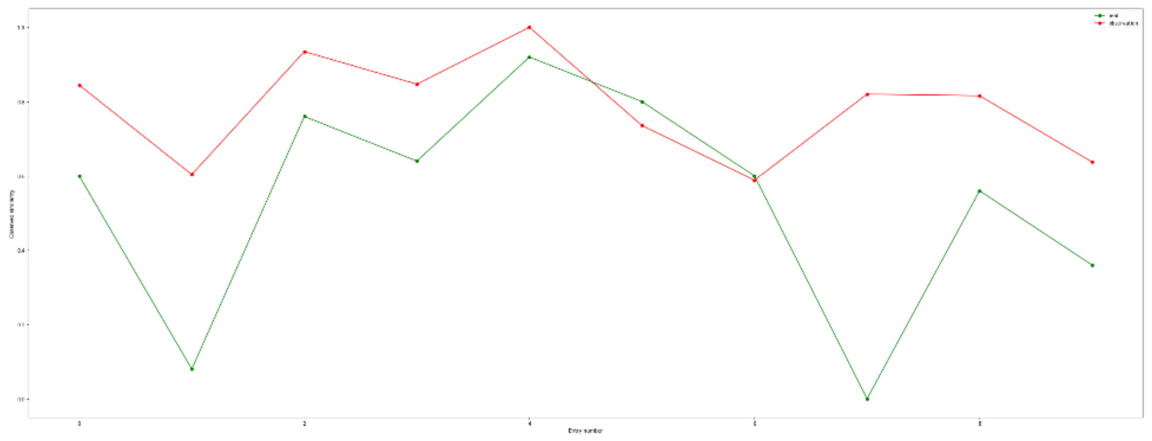
Experimental Results	
Average Difference between Expected and real Comparison Metrics	0.250725989818573
Variance of this difference Comparison Metrics	0.06043110220801425

In the subsequent quarters, we are going to fix this issue by training our embedders with text corpus specific to any domain allowing for better contextual comparison between texts.



X axis : Sentences
Y axis : Difference between observed and real semantic similarity.

This graph shows the difference between read and calculate semantic similarity. For most of the query there is a negative difference, with low variance which indicates that the information captured is more aligned and now even though there is a difference in metric, the measures are aligned toward a particular side.



X axis : Sentences, **Red**: calculated Semantic Similarity, **Green**: Real Semantic Similarity.
Y axis :Plot of observed and real semantic similarity.

This plot gives a comparison of Real Semantic Similarity (Green) and Calculated Semantic Similarity (RED). Here it is observable that for the most part this metric works pretty fine for short queries but for long queries or sentences with synonyms, this metric fails.

Future Plans

For 3rd quarter

On Development side

1. Creation of routes, db layer, an user interface

On the Comparison Engine side.

1. Experimenting with transformers, and fine tuning them for our use case.
2. Creating server side scripts for clients to train on their dataset to provide better results.

For 4th quarter

1. Engine and Server side script integration.

References

- Pennington, J., Socher, R. and Manning, C.D., 2014, October. (Pennington et al. 14)
- Chandrasekaran, D., & Mago, V. (2020). Domain Specific Complex Sentence (DCSC) Semantic Similarity Dataset. *arXiv preprint arXiv:2010.12637*.
- SICK Dataset (Marelli et al. 2014)