# Semantic Similarity Detection

**A BTP Report By**

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

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY**

16 Nov 2021


**2nd 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 Nov 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.

16 November 2021

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.

# Motivation for this 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.

$$\text{Cos}(x, y) = x \cdot y / ||x|| * ||y||$$

- **x** · **y** = product (dot) of the vectors 'x' and 'y'.
- ||**x**|| and ||**y**|| = length of the two vectors 'x' and 'y'.
- ||**x**|| * ||**y**|| = scalar product of magnitude of 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

# Sentence Embedding Using Siamese-SBERT Network

## BERT

**BERT** (Devlin et al., 2018) is a pre-trained transformer network (Vaswani et al., 2017), which sets for various NLP tasks new state-of-the-art results.

The input for BERT for sentence-pair regression consists of the two sentences, separated by a special [SEP] token. **Multi-head attention** over 12 (base-model) is applied and the output is passed to a simple regression function, **Feed Forward NN**, to derive the final label.

A large disadvantage of the BERT network structure is that no independent sentence embeddings are computed, which makes it difficult to derive sentence embeddings from BERT.

We use the pre-trained BERT network and only fine-tune it to yield useful sentence embeddings.

## Sent-BERT or SBERT

**SBERT** adds a **pooling operation** to the output of BERT to derive a fixed sized sentence embedding. For our use case, we are using **MEAN** pooling Strategy.

In order to fine-tune BERT we create **siamese and triplet networks** (Schroff et al.,2015) to update the weights such that the produced sentence embeddings are semantically meaningful and can be compared with cosine-similarity.

We are using **Siamese Network** to train our model

In our old model , we utilized **Regression Objective Function (ROF)** . In ROF, The cosine similarity between the two sentence embeddings u and v is computed. We used Mean-Squared-Error loss as the objective function.

In our Current model, we are using the **Triplet Loss Function.**

# **Siamese Network** and **Triplet Loss Function**

## Siamese Network

A Siamese Neural Network is a class of neural network architectures that contain two or more identical subnetworks. 'identical' here means, they have the same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks. It is used to find the similarity of the inputs by comparing its feature vectors, so these networks are used in many applications

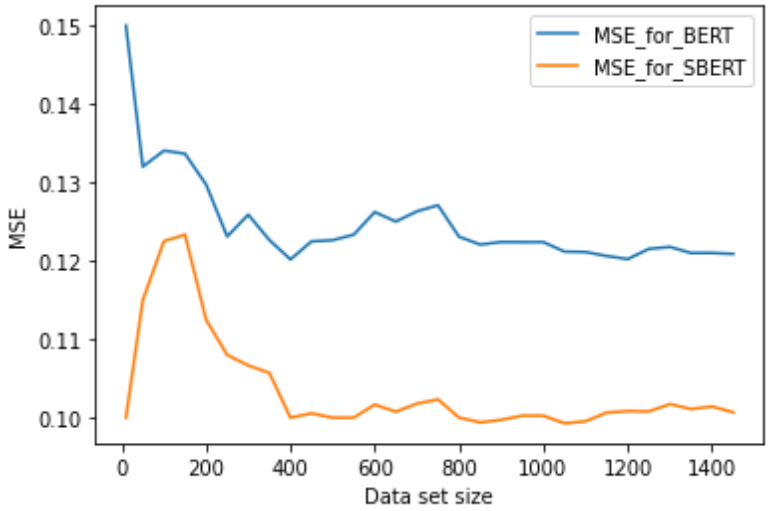
## Triplet Ranking Loss

Given an anchor sentence a, a positive sentence p, and a negative sentence n, triplet loss tunes the network such that the distance between a and p is smaller than the distance between a and n. Margin $\epsilon$ ensures that $\mathbf{s_p}$ is at least $\epsilon$ closer to $\mathbf{s_a}$ than $\mathbf{s_n}$.

$$max(||s_a - s_p|| - ||s_a - s_n|| + \epsilon, 0)$$

# Experimental Results

Observed Difference between **Mean Square Error** from test set between SBERT trained on **ROC loss** and **Triplet loss**, over different data sizes is as below.

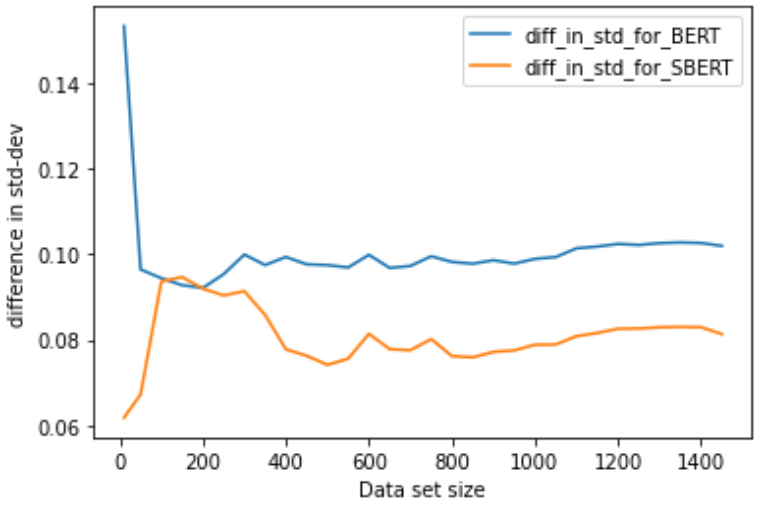



On test set of size 1500 pairs, observed **MSE** for SBERT with

1. ROC Loss : 0.12088

2. Triplet Loss : 0.10068

Observed Difference between **Standard Deviation** from test set between SBERT trained on **ROC loss** and **Triplet loss**, over different data sizes is as below.




On test set of size 1500 pairs, observed **STD-DEV** for SBERT with

1. ROC Loss : 0.10196

2. Triplet Loss : 0.08135

# Product Preview

## Front End

Front end application is developed utilizing React Js for web. This web application provides users with a dashboard displaying their past activities and transaction details. We provide a special web based component to interact with the comparison engine and get quick results.

On the comparison pags we provide a web based visual system to input sentences and get the results instantly. We provide features to compare using currently imputed sentences and features to load saved sentences for users which users can save for their convenience. We also provide options to download a detailed comparison report of the result found in this section itself.

## Back End

Since our focus is a rest based service, we have provided robust documentation to help our customers. We provide every feature, from registration to uploading data to comparison through REST api. List of services available is a below:

1. Login and Registration
2. Get complete user detail and history
3. Upload data for frequent and fast usage
4. Get list of saved sentences
5. Update sentence and related information
6. Custom comparison of sentences ( completely unseen sentences )
7. Add credits to user account

For our customers, we provide the feature to upload sentences and some related information in terms of string with it. This allows for the users to add quick information and codes associated with the sentence, if in case the sentence surfaces, actions can be executed quickly saving them the hashel to manually decide for each sentence every time they receive it.

## Link To Project Repository

1. Server Side : https://github.com/hrs2203/sent_sim_django
2. Client Side: https://github.com/hrs2203/sent-sim-ui

# References

1.. Reimers, N. and Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084.

2. Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

4. Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering.arXiv preprint arXiv:1503.03832, abs/1503.03832.

5. https://github.com/huggingface/transformers ( hugging face, open-source, Transformers Implementation and Base BERT Models )