# CSI5180 Topics in AI: Virtual Assistants
# Winter 2023

# ASSIGNMENT 2
# Building a paraphrase classifier

**Submitted To:**

**Caroline Barrière**

**Team Members:**

1. Abhisht Makarand Joshi – 300311048
2. Nidhi Kumari Chauhan – 300279256

## 1. Introduction

The binary classification of paraphrases is a technique for paraphrase detection that will produce a result that will indicate whether the two sentences are paraphrased or not. To put it another way, it's a method for determining if two or more phrases convey the same thought but use different vocabulary.

Link between paraphrase detection and intent detection:

The goal of phrase detection is to determine whether multiple sections of text, despite differences in wording and phrasing, express the same or comparable meaning. While user input can be conveyed in a variety of ways, intent detection seeks to determine the actual intention or objective behind the user's input.

## 2. Dataset

For the purpose of this assignment, SemEval-PIT2015[1] was used. The dataset was used for an international competition at SemEval 2015, called Paraphrase and Semantic Similarity in Twitter.

### 2.1 Part of Dataset used

We are using all three Train Set (13063 sentence pairs), Dev Set (4727 sentences pairs) and Test Set (972 sentence pairs) for this assignment. Each set contains topic id, topic name, pair of sentences, label, POS, and tags of two sentences. All the 3 sets have a label that suggests whether these pair of sentences are paraphrased or not. So, we have used 3 columns of all the 3 sets that are [‘S1’. ‘S2’, ‘Label’].

### 2.2 Method for changing from graded evaluations to binary:

Dev set has Labels in (X, Y) format where X is Yes (Paraphrases), and Y is No (Not Paraphrases). It is suggested that labels (3, 2), (4, 1), (5, 0) as paraphrases, we labeled them as ‘1’, labels (1, 4), (0, 5) as non-paraphrases labeled as ‘0’, and labels (2, 3) are debatable. So, we dropped all the debatable cases with label (2,3) i.e., (2 Yes, 3 No) in Dev Set. Also, we discarded the data in the Test Set where the label is 3 (Debatable).

Some Examples of Paraphrases and Non-Paraphrases are shown in Table Below:

| Sentence 1 | Sentence 2 | Label | Paraphrase |
|---|---|---|---|
| A Walk to Remember is the definition of true love | A Walk to Remember is the cutest t hing | (3,2) | 1 - Yes |
| Do not Amber Alert me | Just found out Amber was eliminate d | (0,5) | 0 - No |
| Backstrom hurt himself durin g warmups | Backstrom is out we lose | (1,4) | 0 - No |
| Yay my pandora hours reset | I finally get more pandora hours | (5,0) | 1 - Yes |

### 2.3 Subjectivity of the class and the agreement between the 5 judges:

The SemEval-PIT2015 dataset's subjectivity of the class refers to the fact that each item is given a positive or negative rating. Because it depends on the judgements and the amount of experience and competence each judge has, the labelling effort needed may be arbitrary. Although the majority vote technique and the employment of numerous judges can help to minimize personal biases, it is still possible that the decisions will not always be accurate or widely accepted. The majority of the time judges agree like in some instances, when all 5 judges agree about whether a sentence is a paraphrase or not paraphrase, there is a high degree

of agreement. For Example: In Train set, Number of (0, 5) labels are 5212, Number of (5,0) labels are 1247, these are the cases where all the 5 judges agree. In our opinion. We agree with the judges.

## 3. Algorithms/Methods

### 3.1 Baseline:

For the baseline model we used Cosine Similarity, Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them [2]. Regardless of the size of the documents, Cosine similarity is a metric used in NLP to gauge how similar they are. For calculation of Cosine similarity, First, each sentence is represented as a vector. Once the sentence is represented as vectors, we calculate the cosine similarity between two vectors. After calculating Cosine similarity, we compare the cosine similarity with the threshold, if the cosine similarity is greater than the given threshold, the baseline algorithm will consider this pair of sentences as paraphrases otherwise as non paraphrases. Threshold was considering using the mean of the Cosine Similarity.

**For Example:**

S1: A Walk to Remember is the definition of true love
S2: A Walk to Remember is the cutest thing
Cosine Score: 0.447214
If Cosine Score> Threshold:
    Paraphrase: “YES”
    Else:
    Non-Paraphrase: “NO”

**Code snippet for Threshold.**

[24]:
```
def cosine_baseline_threshold(df):
    X=df['Baseline_Cosine_Similarity']
    after_threshold=[]
    for i in range(len(X)):
        if X[i] >= 0.38:
            after_threshold.append(1)
        else:
            after_threshold.append(0)
    df['Baseline_Threshold_CS']=after_threshold
    return df
```

### 3.2 Algorithm A:

In the second algorithm, each pair of words is converted into Term Frequency-Inverse Document Frequency (TF-IDF) [4] vectors, after which the cosine similarity is calculated on each sentence.

It combines two metrics:

I) Term Frequency (TF) is the ratio of a term's (word's) frequency to the total number of words in a document.
II) IDF: The term "inverse document frequency" (IDF) refers to a logarithmically weighted inverse fraction of the number of documents that contain a given word. It is calculated by dividing the total number of documents by the number of documents that contain a given term, and then taking the logarithm of the resulting quotient.

The result of combining TF and IDF for each phrase in a document is then known as TF-IDF. A phrase in a text is considered more important to that document if its TF-IDF score is higher.

$$TF - IDF\ (t, s) = TF\ (t, s) * IDF(t)$$

It calculates the cosine similarity between the two vectors obtained in the preceding step. The two sentences are paraphrases if their cosine similarity is greater than a predetermined threshold. If not, they are not. The Threshold value that is considered was taken using the mean of the Cosine Similarity of TF-IDF.

**For Example:**

S1: A Walk to Remember is the definition of true love
S2: A walk to remember is on ABC family youre welcome
TF-IDF Vector
Cosine Score of (TF-IDF): 0.291219
If Cosine Score of (TF-IDF) > Threshold:
Paraphrase: “YES”
Else:
Non-Paraphrase: “NO”

**Code Snippet:**

```
def TF_IDF_cosine_similarity(sent1,sent2):
    sentence_list = [sent1,sent2]
    vect = TfidfVectorizer(stop_words="english")
    tfidf = vect.fit_transform(sentence_list)
    pairwise_similarity = tfidf * tfidf.T
    cosine_similarity = pairwise_similarity.toarray()[0][1]
    return cosine_similarity
```

**3.2 Algorithm B:**

Sentence-Bidirectional Encoder Representations from Transformers, often known as SBERT, is a category of language representation model that discovers how to encode sentences into fixed-length vectors while maintaining their semantic meaning. Sentence embedding is a technique that does this by converting each sentence to a high-dimensional vector space in which the distances between the vectors reflect the degree of similarity between the respective sentences. Cosine Sentence embeddings are checked for similarity, and if the similarity exceeds a certain threshold, the sentence is paraphrased. Otherwise, they are not paraphrased.

SBERT is based on the well-known BERT architecture, a deep neural network model that has been pre-trained on a sizable corpus of text and may be tailored for a range of subsequent natural language processing tasks, including question-answering, text classification, and text synthesis.

**For Example:**
S1: My first podcast drops on iTunes
S2: the new cd from RRBChoir is now available on iTunes
Cosine Score (Embeddings from Sbert): 0.25345775
If Cosine Score of Cosine Score (Embeddings from Sbert) > Threshold:
Paraphrase: “YES”
Else:
Non-Paraphrase: “NO”

**Code Snippet:**

```
#Compute cosine-similarities using embeddings
sbert_model = SentenceTransformer('sentence-transformers/all-mpnet-base-v2')
def compute_sbert_embedding(df):
    sent1,sent2=Sent1Sent2_list(df)
    sentence_embeddings1 = sbert_model.encode(sent1,show_progress_bar=True)
    sentence_embeddings2 = sbert_model.encode(sent2,show_progress_bar=True)
    return sentence_embeddings1,sentence_embeddings2
```

## 4. Results

For the purpose of this assignment, we have 4 evaluation metrics, Accuracy, Precision, Recall and F1 Score.

- *Comparative Analysis of 3 methods (Baseline, Algo A-TF-IDF, Algo B - Sbert)*

| | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Baseline - Cosine Similarity | 0.66 | 0.51 | 0.65 | 0.57 |
| Algo A – TF-IDF | 0.67 | 0.53 | 0.59 | 0.56 |
| **Algo B – Sbert** | **0.68** | **0.54** | **0.68** | **0.59** |

- *Algo B – Performance Measures of Sbert with Cosine Similarity on Test Data*

| | **Accuracy** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|---|
| **Algo B – Sbert** | 0.73 | 0.42 | 0.70 | 0.53 |

- *Result Discussion*

Among the three algorithms, Algorithm B i.e. **Sbert with Cosine Similarity** performed the best, proving by achieving the highest accuracy, precision, recall and F1-Score. However, Algorithm A, i.e TF-IDF with cosine similarity has also performed good with accuracy of 0.67 while baseline technique of Cosine Similarity achieved 0.66 accuracy which is equally good. Since, the Sbert algorithm is the most advanced algorithm among the 3 methods, therefore it is expected to get the best result. Moreover, the three algorithms has a common technique of calculating the distance of the two sentences i.e cosine similarity but the difference lies in the conversion into vectorization/embeddings of sentences. The most advance is through the Sbert, therefore the sbert should possess the highest accuracy.

*Examples of Algo B – Sbert with Cosine Similarity which got correct/incorrect classified:*

| **Sentence 1** | **Sentence 2** | **Cosine Similarity** | **Algo B: Sbert Result** | **Label** | **Correctly Classified** |
|---|---|---|---|---|---|
| end 8 mile fav part whole movi | rabbit 8 mile place determin make | 0.3476581 | No | No | Yes |
| sarah palin game l ook sexi | sarah palin game i m | 0.7648779 | Yes | Yes | Yes |
| ball dont lie zbo | zbo didnt make shot | 0.5939386 | Yes | No | No |
| end 8 mile fav part whole movi | last 3 battl 8 mile shit | 0.55335885 | Yes | No | No |

## 5. Conclusion

In conclusion, we have conducted an experimentation on building a paraphrase classifier, where we have applied three algorithms to classify if the two sentences are paraphrased or not. However, Cosine similarity has been used in all the three techniques to calculate the distance based on which, it is determined whether the sentences are paraphrased or not. The three algorithms applied are Cosine Similarity, TF-IDF with cosine similarity, and Sbert with Cosine Similarity in which cosine similarity is considered as the baseline algorithm. The best result is given by the Sbert Algorithm pertaining the highest accuracy of 0.68, whereas the baseline model has the lowest accuracy of 0.66.

**Future Scope:** Paraphrasing is a tricky problem, therefore, lots of factors needs to be considered while building a paraphrase classifier. Moving forward, other distances like jaccard distance, *Levenshtein distance* can be considered. Along with it, more deep- learning and NLP algorithms can be discovered with embeddings and vectorization models like SimCSE, Doc2Vec, Word2Vec, etc. Furthermore, parameter hyper tunning on Bert model can be studied.

While paraphrase detection focuses on determining whether the meaning of two phrases is the same or similar, Intent detection involves identifying the underlying intention or purpose behind a user's input. This means that the language and context of the sentence pairs used for paraphrase detection may not be representative of the types of sentences that are typically used for intent detection. Intent detection sentences are more diversified in linguistic features, vocabulary, and grammar to capture the variety of ways users convey their intent. For Example, A user may say "I want to order pizza," "Can you bring me some pizza," or "I'm craving pizza right now" to request food. These sentences vary in vocabulary, grammar, and formality. As a result, the sentences in a dataset for paraphrase detection may have more of the same language features, including vocabulary, grammar, and syntax. Therefore, sentences in this dataset are not meant for an intent detection task.

GitHub Link for Code: https://github.com/NidhiKchauhan/CSI5180-Building-a-parapharse-classifier

## 7. References


[1] Cocoxu. (n.d.). *Cocoxu/semeval-PIT2015: Data and scripts for the shared task "Task 1: Paraphrase and semantic similarity in Twitter (pit)" at semeval 2015*. GitHub. Retrieved March 2, 2023, from https://github.com/cocoxu/SemEval-PIT2015/

[2] Wikimedia Foundation. (2023, February 26). *Cosine similarity*. Wikipedia. Retrieved March 2, 2023, from https://en.wikipedia.org/wiki/Cosine_similarity

[3] Radečić, D. (2022, March 14). *Calculating string similarity in Python*. Medium. Retrieved March 2, 2023, from https://towardsdatascience.com/calculating-string-similarity-in-python-276e18a7d33a

[4] *Sklearn.feature_extraction.text.TfidfVectorizer*. scikit. (n.d.). Retrieved March 2, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[5] *Sentencetransformers documentation*¶. SentenceTransformers Documentation - Sentence-Transformers documentation. (n.d.). Retrieved March 2, 2023, from https://www.sbert.net/

[6] Xu, W., Callison-Burch, C., & Dolan, W. B. (n.d.). *Semeval-2015 Task 1: Paraphrase and semantic similarity in Twitter (pit)*. ACL Anthology. Retrieved March 2, 2023, from https://aclanthology.org/S15-2001/