**Project Report**

### Team – Phoenix

### Navaneetha Krishnan Muralidhar

### Mrugapphan Kannan

# Problem Description:

Design and implement a model to determine Semantic Text Similarity (STS) between two sentences. The similarity score takes an integer value between 1 and 5 (included).

## Data Description:

In the train file four features were present.

* Sentence ID
* Sentence 1
* Sentence 2
* Similarity

The file had 1484 such pairs. Similarly, the validation file used for tuning had 1209 sentence pairs. The test set which has 750 sentence pairs didn’t have the gold tags (similarity score) in it.

# Tools used:

* NLTK - For WordNet, work tokenising, sentence tokenising, POS tagging etc.
* Spacy - To get dependency parse trees.
* NumPy – Similarity between feature vectors.

# Proposed Solution:

## Semantic Similarity between two sentences:

For the given sentence pair, a vocabulary that contains union of tokens in both sentences is formed. Two feature vectors of size equal to the vocabulary was initialized with zeros corresponding to each sentence. For each vector if the word in the vocabulary is present in the sentence then value of the corresponding index is set to one else the value is set to max similarity which is calculated by finding the similarity between each word in the sentence and the present word in the vocabulary. Finally, the dot product between the two vectors is a measure of similarity.

Function name: semantic\_similarity(sentence\_1, sentence\_2)

Input: Pair of sentences: sentence 1, sentence 2.

Output: Similarity score between 1 and 5.

## Path length between two Synsets:

For any two Synsets we find the number of edges in the tree present in the WordNet. This is wrapped around an exponential decreasing function to get a value between zero and one.

This is similar to the WordNet’s path similarity function.

Function name: find\_length(synset\_1, synset\_2)

Input: Two synsets corresponding to two words after WSD.

Output: Real number between zero and one.

## Lowest common ancestor:

For any two synsets we find the lowest common ancestor for the corresponding tree in the WordNet. This is wrapped around an exponential decreasing function to get a value between zero and one.

It is similar to WordNet’s WUP similarity.

Function name: lca\_depth(synset\_1, synset\_2)

Input: Two synsets corresponding to two words after WSD.

Output: Real number between zero and one.

## Word similarity:

It is a function of product of the lowest common ancestor and the path length between the synsets.

Function name: calculate\_word\_similarity(word\_1, word\_2)

Input: Two tokens.

Output: Number between zero and one.

## Similarity based on POS tags:

From the given sentence pair all possible nouns, verbs, adjectives and adverbs are extracted. The values in each of these label belonging to the two sentences are compared using word similarity and a score is generated for the sentence pair between a value of one and five.

## Helper methods:

### Extract synsets:

This function compares the synsets of the two words and returns the most relevant synset pair of the lot.

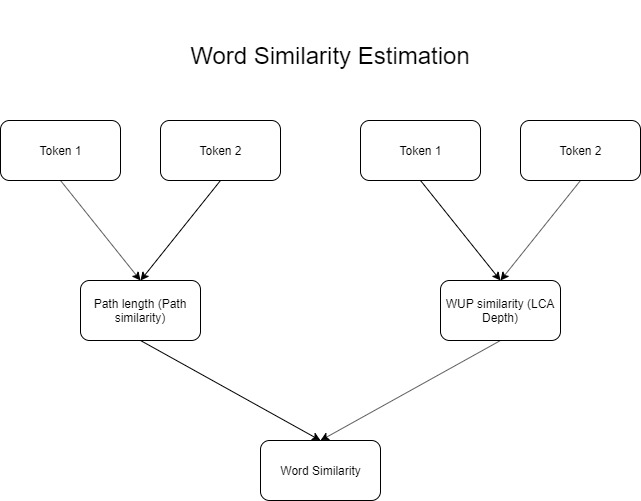
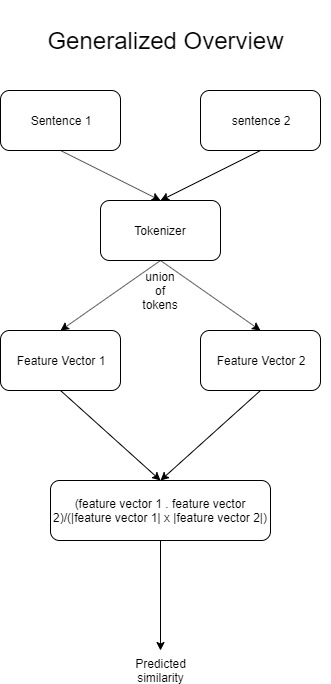
### Hypernym distance:

Returns a dictionary of hypernyms as keys and distance of the hypernym from the current word as value.

### Similarity Estimate:

The results from both POS tagging based approach and semantically evaluating similarity are combined using a linear function to output an integer number between 1 and 5.

Based on the accuracy in development set, a weight of 0.6 is assigned to semantically evaluating similarity and 0.4 is given to POS tagging based approach.



### Error Analysis

Both the statistical based approached gave out real numbers with 7 significant decimal places. In order to follow up with the pattern supported by evaluation.py, the real numbers were converted to integers by rounding. This resulted in considerable amount of drop in accuracy obtained.

### Samples affected by rounding

sentence\_1 = "The support will come as a free software upgrade called WebVPN for current customers that have support contracts."

sentence\_2 = "The upgrade will be available as a free download for current customers with SmarNet support in January 2004."

semantic\_similarity(sentence\_1, sentence\_2) = 3 (rounded down from 3.3) -> True label = 4

There were also a considerable amount of samples which gained by rounding but the fraction was less when compared to the converse.

### Potential Improvements

The extracted statistical features could be fed in to machine learning based model instead of combining them linearly. This may lead to a better accuracy metric.

Identification of a more accurate metric to measure word similarity between proper nouns.