# CS 6320 – Natural Language Processing

# Project Report

# Problem Description:

The project deals with designing and implementing a system that determines how similar two chunks of text are. The similarity between these chunks is determined by assigning an integer value between 1 and 5 (included). The higher the score, the more similar these two chunks are.

The following are some sentences and their scores to get an understanding of how Semantic Textual Similarity gives similarity scores.

**Sentence1**:Counties with population decline will be Vermillion, Posey and Maddison

**Sentence2**:Vermillion, Posey and Maddison county population will decline.

**Score:** 5

**Explanation:** Both sentences convey the same meaning of population decline in particular counties.

**Sentence1**: They were at Raffles hospital over the weekend for further evaluation.

**Sentence2**: They went more tests over the weekend and are now warded at Raffles Hospital.

**Score**: 3

**Explanation**: Although both the sentences mention the word Raffles hospital, sentence2 is providing more information about what happened after sentence1 in time.

**Sentence1**: Hilsenrath and Klarman each were indicted on three accounts of security fraud.

**Sentence2**: Klarman was charged with 16 counts of wire fraud

**Score**: 1

**Explanation**: Even though both the sentences contains Klarman, his actions are different.

# Proposed Solution:

With inline train-set corpus being added in to an object, each paraph (consisting of two sentences’ hashmaps and score class) is subjected to process through famous Natural Language Processing (NLP) libraries namely spacy and wordnet. Various features which will be discussed going forward are extracted with respect to these two-sentences and trained with machine learning models. The three models we used are Random Forest, Support Vector Machine and Adaboost.

The test-set/dev-set corpus is proposed to go through the same procedures as above to extract those exact features as their trained set. The trained machine learning models then are used to classify the test-set/dev-set s’ data with the help of these features.

Out of the three classifiers’ output, the class with high frequency is chosen although weightage is given in case of none-determination.

# Features extracted:

Longest subsequence: Each word of the two sentences are changed to their lemma selves. And then these two sentences are subjected to find the longest subsequence which is later normalized with sum of lengths of both sentences

LCS = length of subsequence/sum(length of string1, length of string2)

Jaccard Distance: Jaccard distance of two sets is defined as the ratio of number of common elements of both sets to number of elements in union of both sets. Here the Jaccard distance is calculated with respect to sets of two sentences containing lemmas

JD = number of common elements in A and B/ number of all elements in A and B

Lavenshtein Distance: Lavenshtein distance of two strings is defined as number of edit operations such as insertions, deletions and substitutions of words need to be made in order to change from string1 to string2.

Parts of Speech overlap features: The main parts of speech of any given sentence are nouns, verbs, adjectives and adverbs. There is a lot of information which can be deduced from these hence Jaccard distances of these sets are calculated in the field of parts of speech features

Spacy Similarities: Dependency parsing is one of the main techniques used to find sentence similarities and here we use the parsing to bag the nsubj, pobj and dobj s’ in order to find their Jaccard Similarities as features

## POS based distance metric:

When given two sentences we extract nouns, verbs, adjectives and adverbs into separate sets for each sentence. Then the following features are extracted:

npos- jaccard distance between two sets containing nouns from each sentence.

vpos - jaccard distance between two sets containing verbs from each sentence.

apos - jaccard distance between two sets containing adjectives from each sentence.

rpos - jaccard distance between two sets containing adverbs from each sentence.

## NGram Overlap:

When given two sentences both bigrams and trigrams are extracted for each sentence. Then the ‘bigramOverlap’ score and ‘trigramOverlap’ score is calculated by the following harmonic mean formula:

Score =

Where |A| is the size of set A (either bigram or trigram set of sentence A), A ∩ B is the intersection of two sets (either bigram or trigram) and |B| is the size of set B (either bigram of trigram set of sentence B).

## Word Similarity Score:

Say given two sentences for similarity score, we extract three features called ‘nsubSimilarity’, ‘pobjSimilarity’, ‘dobjSimilarity’. For that, first we extract nsubj, pobj and dobj tokens of both sentences into separate sets of each type using dependency parse tree.

nsubj1 = {all nsubj of sentence1}, nsubj2 = {all nsubj of sentence2}, pobj1 = {all pobj of sentence1},

pobj2 = {all pobj of sentence2}, dobj1 = {all dobj of sentence1}, dobj2 = {all dobj of sentence2}

Now for each nsubj in nsubj1 we calculate the path similarity score between every nsubj in nsubj2 using wordnet and take the max value. So, we get nsubj scores of length nsubj1. We take the mean of these values and will be called ‘nsubSimilarity’ score. Through the same procedure we get ‘pobjSimilarity’ and ‘dobjSimilarity’ scores.

# Programming Tools:

## NLTK:

NLTK is a leading platform which provides easy-to-use interfaces for lexical resource such as wordnet. We have imported and used lesk for word sense disambiguation. The tokenization of words and finding the appropriate pos tags for each word in a sentence is done using ‘word\_tokenize’ and ‘pos\_tag’ imports respectively.

## Spacy:

Spacy is a free open source library for NLP which can be used to preprocess or extract useful information from the text. We have loaded ‘en\_core\_web\_md’ model to extract features like tokens, pos, lemma and dependency parse tree from each sentence for our NLP tasks.

## Pandas:

Pandas is a python language software library used for data manipulation and analysis. It offers data structures like dataframe which can be used for storing and manipulation of data. In our project we have used dataframe to store all the extracted features for each of our sentence. The same dataframe is also used to train our machine learning algorithms.

## Scikit-Learn:

Scikit learn is an free open source machine learning library for the python programming language. We have imported cosine\_similarity to calculate the cosine similarity of two vectors. The machine learning algorithms such as Random Forest, SVM and Adaboost classifier that are used to build our STS system are all imported from scikit-learn.

A screenshot of a cell phone

Description automatically generated

Fig. 1.1 Architectural Diagram of Proposed Solution

# Results and Error analysis:

The results were astounding and far better than that of a random classifier. With a consistent person coefficient score of 0.446, the model which we developed seemed to be of a high academic standard example.

Good examples here imply how efficient the model working is. It is clearly shown with spelling bee example where those two sentences almost match up with little variation in the information contained. Hence it has been classified to a score class 4.

Variant examples show that the model which we trying are focusing on subject, object and the relation between them and less focusing on numerical/objective values they are bound with. It can be clearly seen in both examples how the model was falsely assuming the sentences are in certain way similar.

## Good Examples:

s\_500

Sent1: Tom Kasmer, a 14-year-old from Belmont, N.C., got a word that sounded like "zistee" during yesterday's competition.

Sent2: The 14-year-old national spelling finalist who attends school in Belmont, N.C., got a word that sounded like "zistee" during competition Wednesday.

Score: 4

Predicted Score: 4

---

s\_535

Sent1: Aspen Technology's shares dropped 74 cents, or 23 percent, to close at $2.48 on the Nasdaq.

Sent2: In afternoon trading, Aspen's shares were off 89 cents or more than 27 percent at $2.33 per share.

Score: 2

Predicted Score: 2

## Variant Examples:

s\_505

Sent1: Earlier this month, RIM had said it expected to report second-quarter earnings of between 7 cents and 11 cents a share.

Sent2: Excluding legal fees and other charges it expected a loss of between 1 and 4 cents a share.

Score: 1

Predicted Score: 4

---

s\_543

Sent1: Analysts expected earnings of 27 cents a share on revenue of $17.7 billion, Thomson First Call says.

Sent2: Hewlett-Packard is putting in for a second-quarter profit of $659 million, or 29 cents a share, on revenue of $18 billion.

Score: 1

Predicted Score: 3

# Problems Encountered:

## Using lesk:

Lesk is an algorithm which can be used for word sense disambiguation to get the most probable synset for a word. Lesk takes three parameters as inputs- sentence sent as list of words, word for which we want the synset and pos tag of the word. First two parameters can be easily obtained. For the last one we have tried using nltk.pos\_tag to get the appropriate pos tags. Then we have realized that the format is different. For example, say for noun pos\_tag can give {NN, NNS, NNP, NNPS}. But we can only give ‘n’ as pos tag to the lesk algorithm. So, this problem provoked us to create a dictionary of pos tags in python where the key value will be the pos tags of pos\_tag format and the values will be the pos tags of format which can be given to the lesk algorithm. After obtaining the pos tags from nltk.pos\_tag we map and change the format before passing them to lesk.

## Using Wordnet to check word similarity:

At first, in one of our features we used to obtain nsubj, pobj, dobj of two sentences and measure the distance between them directly. There is a flaw in the feature as it only works efficiently when the words are the same in both the sentences. After a brief discussion with the professor, we have realized that we can extend the feature to create a new one by checking the similarity between two words using wordnet and giving it a score between 0 and 1 rather than just checking whether two words are equal or not.

# Pending Issues:

## Named Entities:

In our present implementation we don’t have any separate feature to capture the similarity of named entities (NE) between two sentences i.e. even though we have features such as least common subsequence, jaccard distance which calculate the distance between two sentences they don’t consider named entities separately. Since NE’s are key to any sentence, a separate feature should be assigned to them.

## Punctuations:

The present implementation of STS system does not take any punctuation characters into consideration. A sentence with punctuation marks and the same sentence without punctuation are treated the same way, which is not the case in the real-world scenario. So, this should be handled to get better results.

# Potential Improvements:

There is always a room for improvement and that is true even in our case. Particularly in the measure of pearson coefficient the max value it can take/get is around 1 and we were getting a consistent value of 0.446.

In case of features extraction there is always novel features which retain large amount of classifier information. And even in the case of classifiers more impounding methods such as Heuristic and Statistical algorithms give results as good as the Machine Learning techniques.

When it comes to Machine Learning, various famous methods such as Naive Bayes and K-Nearest Neighbors can chip in the make the model more efficient.