**PROJECT REPORT**

# MANOJ KUMAR PANCHAGNULA 11126252

**PROJECT TITLE:** Semantic Textual Similarity

**PROBLEM STATEMENT:**

Given two sentences, the task is to determine whether they mean the same and how similar the sentences are.

**DATA SETS:**

For finding the textual similarity different kind of data sets are used,

* A dataset with sentences, which are in active voice and passive voice form. This dataset is used because, when the two sentences are in active and passive voice form then they mean the same, so by finding the textual similarity we must get high similarity measure for the sentences.
* The numerical data is also used to find the similarity measure.

**APPROACH:**

1. **BASELINE APPROACH:**

* Given the text, a baseline approach is that, to calculate the word overlap.
* Initially, given text is preprocessed by removing the special characters from the sentences and converting all the letters to lowercase letters.
* The tokens are extracted from the sentences by splitting the sentences using white space.
* Once we get the two lists with tokens of both the sentences, both the lists are compared and the common words in both the lists are extracted.
* Once we get the common words in both the texts, we find how much percent both the texts are similar by using the formula (len(common words)\*2/len(sentence1 + sentence2) )\*100
* We are scaling the results on a scale of five we multiply the obtain percentage with .05
* For implementing the above procedure, I have implemented a function ‘create\_model():’ which will return me the tokens of the given text by preprocessing the text and splitting the text.
* Once the tokens obtained we will find the similarity of the sentences on the scale of five as explained above.

1. **POS Tagging Approach:**

* To compare the sentences, comparing the common words might not give the high accuracy for the sentences for that, to find the better, comparing their parts of speech tags will give the better results.
* For finding the pos tags for the tokens, I used a library nltk.pos\_tag(token) from nltk toolkit.
* Using that library, I acquired the pos tags for the tokens, of the given sentences.
* Once I got the pos tags, I compared the number of nouns, verbs and adjectives of the tokens are present in the sentences.
* For implementing the above I used a function ‘pos\_tagger():’, which will find the pos tags of the tokens.
* This function takes the input of the tokens from the ‘create\_model():’ function.
* Once it finds the tags for the tokens, we need to find the count of the pos\_tags of the tokens of the sentences.
* For find the count of the pos tags, I used Counter library from colloections package. It helped for finding the count of the pos tags present in the sentence.
* The counter returns a dictionary with keys as tokens and the values as the count the pos tag.
* So the pos\_tagger(): function returns a dictionary with pos tags and their counts.

1. **Finding the similarity:**

* Now we have a dictionary with pos tags and their counts, now for finding the similarity of the sentences, I have written a function ‘Similarity():’, which will calculate the similarity of the pos tags and returns the similarity measure on a scale of five.
* For finding the similarity the formula used is ((count of pos tag1)/(count of pos tag2) ) \* 5

1. **Pos tags used to compare:**

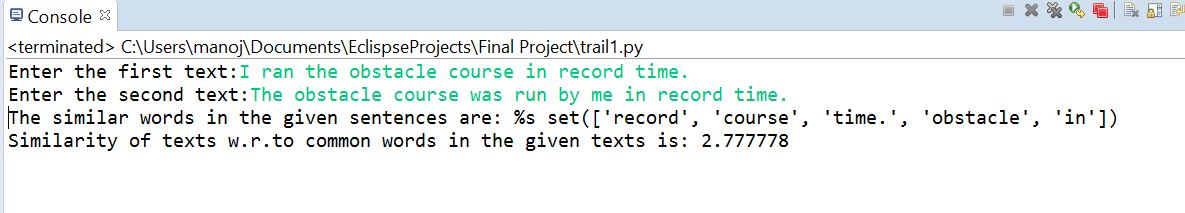
* The pos tags that are used to compare are nouns, verbs and adjectives.
* We have four noun tags, NN, NNP, NNS, NNPS. We need to sum count of all the noun tags and store in a variable. Once we get count for both the sentences, we pass both the variables to the similarity function and get the similarity measure by comparing the noun tags on a scale of five.
* We have four verb tags, VB, VBZ, VBP, VBN. We need to sum count of all the verb tags and store in a variable. Once we get count for both the sentences, we pass both the variables to the similarity function and get the similarity measure by comparing the verb tags on a scale of five.
* In the third comparison instead of finding the similarity of just adjective tags we find the similarity for verb, noun and adjective tags. For adjective tags we have three different tags such as, JJ, JJS, JJR. Now we sum all the noun, verb and adjective tags counts and store them in a variable. Once we find the counts for both the sentences we pass the variables to the similarty function and get a similarity measure on a scale of five.

**Output:**

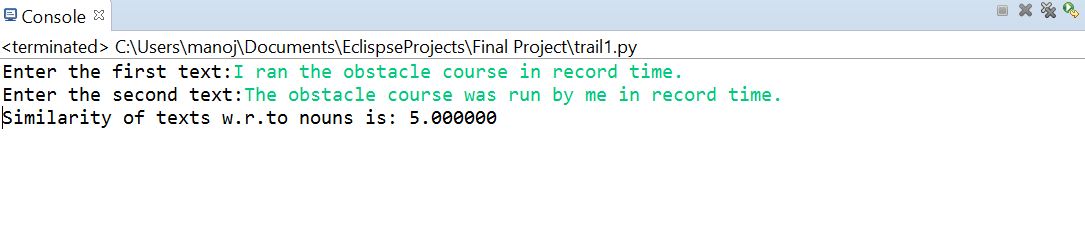
**The sentences use are**

1. I ran the obstacle course in record time.
2. The obstacle course was run by me in record time.

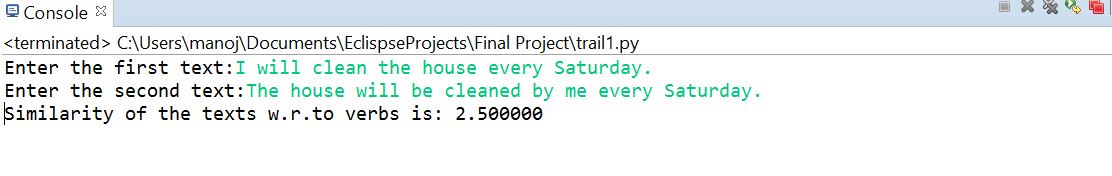
* For a base line comparison



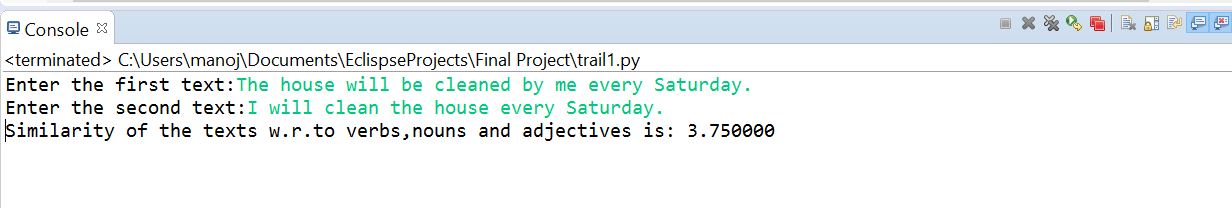
* For comparing the common noun tags,



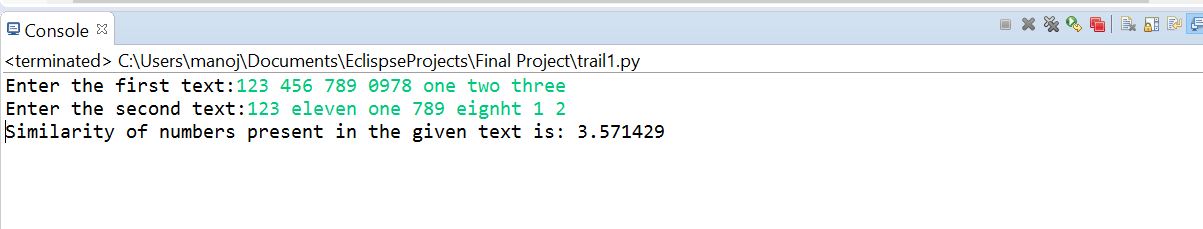
* For comparing the common verb tags,



* For comparing the common verbs, nouns and adjective tags,



* For numerical data,



**PROBLEMS FACED WHILE IMPLEMENTING AND OVERCOMING THEM:**

* The pos tagger function returned me a list with token and it’s pos tags, but the returned list was like (token, tag), it looked difficult for retrieving all the tags and their counts from the list.
* For overcoming the above problem, I used the counter library from the collection package where it returned me a dictionary with tag and its counts.