**Title: Determining semantic similarity of short text using lexical semantics (specifically WordNet) : 10 Marks**

Goal of this assignment is to have a hands on exposure to lexical semantics. You need to mail a python notebook indicating your SRN. Late submission will have a penalty.

**Required background:**

1. **Natural Language Processing 10:** classroom material explaining various wordnet based algorithms for semantic relatedness and Word Sense Disambiguation. This material explains the various algorithms to find semantic similarity between lexemes.
2. **Hands on experience with WordNet API using Python NLTK:** You need to learn wordnet API for semantic similarity between lexemes. You can refer the following notebooks apart from internet resources
    a. Google Drive -> code_used_in_class -> wordnet_NLTK
    b. Google Drive -> code_used_in_class-> Wordnet_based_similarity
    c. Google Drive -> code_used_in_class->NLTK_intro->NLTK_introduction

**Requirement:**

(a) The program should have defined input of pairs of short sentences as below
(b) Should print out similarity value as below (the values given below are just arbitrary representative numbers) but use the same test sentences as below for uniformity :
(c) Don't worry about the exact similarity values you are getting. **You can try out different algorithms (for sense similarity).** Typically the similarity number will also vary based on test sentences.
(d) All words or lexemes or senses may NOT be defined in the WordNet even though it is extensive. You need to handle such cases in your code.

["I like that food.", "That dish is exciting ", 0.561],
["Ram is very nice.", "Is Ram very nice?", 0.977],
["Pure malt whiskey.", "Fresh orange juice.", 0.611],
["It is a dog.", "That must be you!", 0.439],
["It is a cow.", "It is a pig.", 0.623],
["I have a pen.", "I am eating a cake", 0.0],
["I have an identity.", "This is my photograph.", 0.121]

(e ) You should iterate this above step and finally choose the best semantic similarity algorithm as per your test results! What is this algorithm?

**Note:** this assignment is partly adapted from an IEEE paper which suggested a method of finding similarity of short text segments. The actual algorithm uses few other factors in a different form to calculate the semantic similarity.

**How to calculate similarity of short sentences?**

**Step 1: (semantic similarity between words)**

Given two words, w1 and w2, we need to find the semantic similarity. We can do this by analysis of the lexical knowledge base (WordNet). Words are organized into synonym sets (synsets), with semantics and relation pointers to other synsets. Therefore, we can find the first class in the hierarchical semantic network that subsumes the compared words. One direct method for similarity calculation is to find the minimum length of path connecting the two words. Hypothetically speaking, the shortest path between **boy** and **girl** is **boy-male-person-female-girl** and the minimum path length is **4**. The synset of person is called the subsumer for words of boy and girl. Similarly, the minimum path length between **boy** and **teacher** is 6. Thus, we could say girl is more similar to boy than teacher to boy! However, this method may be less accurate if it is applied to larger and more general semantic nets such as WordNet. So, there are various algorithms available and you should try out different algorithms covered in the class.

1. So given two short sentences, you should represent each short sentences by a list of "content words"
2. You should combine these two lists into a single list of content words. This should define your corpus vector.
3. Now you should come up with a nxn matrix of calculated similarity between the words (may be using different algorithms with the help of WordNet)

**Step 2: (semantic vector representing sentences)**

Sentences are made up of words, so it is reasonable to represent a sentence using the words in the sentence.

Given two sentences, T1 and T2, a joint word set is formed **T : T1 U T2**. The joint word set T contains all the distinct words from T1 and T2. Since inflectional morphology may cause a word to appear in a sentence with different forms that convey a specific meaning for a specific context, we use word form as it appears in the sentence and do not normalize them.

Since the joint word set is purely derived from the compared sentences, it is compact with no redundant information. The joint word set, T, can be viewed as possible semantic information for the compared sentences. Each sentence is readily represented by the use of the joint word set as follows:

- **The vector for each sentence $T_i$ derived** from the joint word set is called the **lexical semantic vector $S_i$** which consists of some numbers corresponding to the words in the joint word set $w_{i \text{ where } I = 1,2,3,\ldots m}$ . m is the total number of words in the joint word set.
- The **dimension of semantic vector $S_i$** equals the number of words in $w_i$.
- The value of an entry of the lexical semantic vector is determined by the semantic similarity determined so far. Take T1 as an example:
    - Case 1: If $w_i$ appears in the sentence T1, $S_i$ is set to 1.
    - Case 2: If $w_i$ is not contained in T1, a semantic similarity score is computed between $w_i$ and each word in the sentence T1, using the similarity matrix

already computed. Thus, the most similar word in T1 to $w_i$ is that with the highest similarity score x. If x exceeds a preset threshold, then $s_i$ = x; otherwise, $s_i$ =0. You should choose a value here for x.

- The reason for the introduction of the threshold choice is twofold. First, since we use the word similarity of distinct words (different words), the maximum similarity scores may be very low, indicating that the words are highly dissimilar. In this case, we would not want to introduce such noise to the semantic vector. Second, classical word matching methods can be unified into the proposed method by simply setting the threshold equal to one.

### Step 3: Calculate the similarity

The semantic similarity between two sentences is finally calculated as the cosine coefficient between the two semantic vectors using dot product. Print this value along with the short phrases that you are comparing.

**Other idea that you may like to play with**

If you are comfortable with lexical semantics now, you can learn key phrase extraction from text. Use this knowledge of key phrase extraction to do the same for two articles and come up with semantic similarity between them!

1. First extract key phrases from each article
2. Then use the above method to do a mxn similarity matrix between key phrases where m and n are the number of key phrases from each article
3. Each article can now be represented by a vector of key phrases
4. Possibly determine cosine distance to calculate the similarity between two article