Natural language Processing

(UE14CS333)

Assignment II

Continuous text fragment similarity Estimation

By,

Sriharsha Hatwar K.S (01FB14ECS250)

Abhilash K.R (1PI14CS003)

Implementation:

Basically this Assignment is implemented in Python which deals with the Semantic similarity measure between two sentences. The technologies / Libraries used are nltk, wordnet, numpy.

The implementation goes like this, when the user inputs two sentences initially the sentence are word tokenised and then these following steps takes place:

1) We create a joint word set which basically contains all possible words in two sentences and then we create two semantic vector for these two sentences and store in a numpy array, the fields are filled on certain conditions which is explained in the research paper which involves word similarity between to words and these word similarity Is based on the length between two words to be compared and the depth from the 'entity.n.01' to the lowest common subsumer in the wordnet. Then there is a function function(h,(height=depth)) that monotonically increases the value b/w (0,1) when the depth increases and then there is a function f(l) which decreases as the length between two words in the wordnet increases and the range of this function is also between (0,1). Using these functions we f(l) * f(h) we determine the word similarity .

2) After the generation of semantic vectors we do the cosine similarity between two semantic vectors, the result is treated as S_s.

3) Now comes the word order similarity this is to be calculated as the sentence that is to be compared with other sentence might have the same word but in different order and this difference in order might result in slight change of the meaning of the sentence, so by creating two word order vectors we assign the index number to each word order vector from the joint word set and then we do the operation: $|r1 – r2| / |r1 + r2|$ which determines the word order similarity between two sentences represented by S_r.

4) After determining both word order similarity S_r and semantic similarity S_s we use these two to determine the overall similarity by using a constant delta that determines the contribution of each similarity to overall similarity.

Overall similarity (S) = $\delta * S_s + (1-\delta) * S_r$

Some examples of sentence similarity:

Example 1:

1) He is a boy

2) He is a lad

As we know that lad and boy are same the overall similarity measure was equal to: 0.98090068417

Example 2:

1) Sushrith is a good boy

2) I like hockey

As we can see here that these two sentences are completely unrelated , the similarity measure b/w these two is very low : 0.0137946843897

Example 3:

1) He likes hockey

2) He likes hockey

As we can see both sentences are same it is clearly evident that the similarity measure is: 1.0

Important Point: *As a thumb rule if the similarity measure is less that 0.2 then we can clearly say that the two sentences are not at all similar and if it crosses 0.5 we can say that they are almost similar.*

Example 4:

1) He drinks wine

2) He drinks red alcohol.

The similarity measure is: 0.965220258819 as both the drinks refer to alcoholic beverages the measure will be pretty high.

Example 5:

When we input a file containing variety of sentences we first sentence tokenize it and then for each combination of sentences we determine the similarity measure and then store it in a matrix.

Input file : file.txt (Containing 4 sentences)

Output:

```
[[ 1.         0.18953879  0.14296844  0.12067199]
 [ 0.18953879  1.         0.67454448  0.67759147]
 [ 0.14296844  0.67454448  1.         0.52786927]
 [ 0.12067199  0.67759147  0.52786927  1.        ]]
```

Learning Outcomes:

The outcomes that we learnt from this assignment is that in our Adv. Algo Assignments we tried syntax similarity (String matching) using different algorithms, but in this Assignment we learnt how to do the semantic similarity b/w two sentences which takes lots of time when compared to string matching as this uses one of the finest of lexical database known as wordnet. As we wanted matrix operation to be performed using lists is ruled out as it will take extra time in computation, so we used numpy which helps in matrix manipulation faster. Finally we thoroughly enjoyed the assignment as it gave us an experience as how semantic similarity is used in doing information extraction.

Thanking you,

Sriharsha Hatwar K.S

Abhilash K R