Practical no 5

Name : Shivam Tawari .
Roll no: A-58
Subject: NLP

Aim: Write a python program for distributional matrix to find computational similarity from given dataset.

Theory:

- Distributional Matrix Semantics Model:

→ DSM are the computational model that build contextual semantic representation from corpus data.

→ DSM are model for semantic representation. The semantic context is represented by a vector.

→ Distributed semantics is the study of statistical patterns.

→ Distribution are vectors in multidimensional semantic space that is object with magnitude

and direction.

The semantic space has dimension which correspond to possible context as gathered from a given corpus.

Constructing word spaces :

① Pick the word you are interested in target.

② Defines a context window number of words surrording target word.
The context can be general be defined in terms of documents, paragraph.

③ Count number of times the target words co-occurs with context words co-occurance matrix.

④ Build vectors out of function these co-occurance counts.

Conclusion: Hence, we have successfully implemented the program to find the distributional matrix, computational similarity from given dataset.

Practical – 5

Name: Shivam Tawari

Roll no: A – 58

Code:

NLP Practical 5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

NLP Practical - 5

Name: Shivam Tawari

Roll no: A - 58

```
[11] import nltk
     from nltk.corpus import wordnet
     from nltk.stem import WordNetLemmatizer

     nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
[12] sentence = """An automobile is a wheeled motor vehicle used for transporting passengers.
     A car is a form of transport , usually with four wheels and the capacity to carry around five passengers.
     Transport for the London games is limited , with spectators strongly advised to avoid the use of cars.
     The London 2012 soccer tournament began yesterday , with plenty of goals in the opening matches.
     Giggs scored the first goal of the football tournament at Wembley , North London.
     Bellamy was largely a passenger in the football match , playing no part in either goal.
```

+ Code + Text

RAM Disk Editing

[12] sentence = """An automobile is a wheeled motor vehicle used for transporting passengers.
A car is a form of transport , usually with four wheels and the capacity to carry around five passengers.
Transport for the London games is limited , with spectators strongly advised to avoid the use of cars.
The London 2012 soccer tournament began yesterday , with plenty of goals in the opening matches.
Giggs scored the first goal of the football tournament at Wembley , North London.
Bellamy was largely a passenger in the football match , playing no part in either goal.
"""

```
[14] # Map POS tag to first character lemmatize() accepts
     def get_wordnet_pos(word):
         tag = nltk.pos_tag([word])[0][1][0].upper()
         tag_dict = {"J": wordnet.ADJ,
                     "N": wordnet.NOUN,
                     "V": wordnet.VERB,
                     "R": wordnet.ADV}
         return tag_dict.get(tag, wordnet.NOUN)

     lemmatizer = WordNetLemmatizer()
```

```
[15] prep = []
     for sentence in sentences:
         prep.append([lemmatizer.lemmatize(w, get_wordnet_pos(w)) for w in nltk.word_tokenize(sentence)])

     sentences = prep
```

+ Code + Text RAM Disk Editing

Target Words

```
[16] ## example: automobile, car, soccer, football
     target_words = map(str, input('Target Words: ').split(','))
     target_words = [t_wrds.lower().replace(' ', '') for t_wrds in target_words]

     target_words
```

```
Target Words: soccer, football, car, automobile
['soccer', 'football', 'car', 'automobile']
```

Vocab

```
[17] ## example: wheel, transport, passenger, tournament, London, goal, match
     term_vocab = map(str, input('Term Vocabulary: ').split(','))
     term_vocab = [t_voc.lower().replace(' ', '') for t_voc in term_vocab]

     term_vocab
```

```
Term Vocabulary: wheel, transport, passenger, tournament, london, goal, match
['wheel', 'transport', 'passenger', 'tournament', 'london', 'goal', 'match']
```

+ Code + Text RAM Disk Editing

[17] ## example: wheel, transport, passenger, tournament, London, goal, match
term_vocab = map(str, input('Term Vocabulary: ').split(','))
term_vocab = [t_voc.lower().replace(' ', '') for t_voc in term_vocab]

term_vocab

Term Vocabulary: wheel, transport, passenger, tournament, london, goal, match
['wheel', 'transport', 'passenger', 'tournament', 'london', 'goal', 'match']

```
[18] context_window = 'sentence'
```

```
[20] matrix = {}
     for k in target_words:
         matrix[k] = {}
         for v in term_vocab:
             matrix[k][v] = 0

     for k, v in matrix.items():
         print(k, v)
```

```
soccer {'wheel': 0, 'transport': 0, 'passenger': 0, 'tournament': 1, 'london': 1, 'goal': 1, 'match': 1}
football {'wheel': 0, 'transport': 0, 'passenger': 1, 'tournament': 1, 'london': 1, 'goal': 2, 'match': 1}
car {'wheel': 1, 'transport': 2, 'passenger': 1, 'tournament': 0, 'london': 1, 'goal': 0, 'match': 0}
automobile {'wheel': 1, 'transport': 1, 'passenger': 1, 'tournament': 0, 'london': 0, 'goal': 0, 'match': 0}
```

+ Code + Text RAM Disk Editing

for k in target_words:
[20] matrix[k] = {}
for v in term_vocab:
matrix[k][v] = 0

for k, v in matrix.items():
print(k, v)

soccer {'wheel': 0, 'transport': 0, 'passenger': 0, 'tournament': 1, 'london': 1, 'goal': 1, 'match': 1}
football {'wheel': 0, 'transport': 0, 'passenger': 1, 'tournament': 1, 'london': 1, 'goal': 2, 'match': 1}
car {'wheel': 1, 'transport': 2, 'passenger': 1, 'tournament': 0, 'london': 1, 'goal': 0, 'match': 0}
automobile {'wheel': 1, 'transport': 1, 'passenger': 1, 'tournament': 0, 'london': 0, 'goal': 0, 'match': 0}

```
[22] final = []
     for tgt_wrd1, trm_vocab1 in matrix.items():
         final.append(tgt_wrd1)
         for tgt_wrd2, trm_vocab2 in matrix.items():
             if tgt_wrd2 in final:
                 continue
             similarity = sum([a*b for a,b in zip(list(trm_vocab1.values()), list(trm_vocab2.values()))])
             print(f'{tgt_wrd1} . {tgt_wrd2} = {similarity}')
```

```
soccer . football = 5
soccer . car = 1
soccer . automobile = 0
football . car = 2
football . automobile = 1
car . automobile = 4
```