CO summarization_2.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM
Disk

Editing

```python
[22] import nltk
     from nltk.corpus import stopwords
     import csv
     from nltk.tag import pos_tag # for proper noun
     from nltk.tokenize import word_tokenize, sent_tokenize
     import pandas as pd
     import math
     from nltk.stem import PorterStemmer
```

```python
[39] filename="/006.txt"
     f = open((filename), "r")
     text=f.read() #append each line in the file to a list
     f.close()
```

```python
[40] nltk.download('punkt')
     nltk.download('averaged_perceptron_tagger')
     nltk.download('stopwords')
```

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

```python
[41] sent_tokens = nltk.sent_tokenize(text)
     word_tokens = nltk.word_tokenize(text)
     word_tokens_lower=[word.lower() for word in word_tokens]
     stopWords = list(set(stopwords.words("english")))
     word_tokens_refined=[x for x in word_tokens_lower if x not in stopWords]
     print(len(word_tokens_refined))
```

```
244
```

```python
[42] stem = []
     ps = PorterStemmer()
     for w in word_tokens_refined:
         stem.append(ps.stem(w))
     word_tokens_refined=stem
```

```python
# .....................part 1 (cue phrases).................
QPhrases=["incidentally", "example", "anyway", "furthermore","according"
          "first", "second", "then", "now", "thus", "moreover", "therefore", "hence", "lastly", "finally", "summary"]

cue_phrases={}
for sentence in sent_tokens:
    cue_phrases[sentence] = 0
    word_tokens = nltk.word_tokenize(sentence)
    for word in word_tokens:
        if word.lower() in QPhrases:
            cue_phrases[sentence] += 1
maximum_frequency = max(cue_phrases.values())
for k in cue_phrases.keys():
    try:
        cue_phrases[k] = cue_phrases[k] / maximum_frequency
        cue_phrases[k]=round(cue_phrases[k],3)
    except ZeroDivisionError:
        x=0
print(cue_phrases.values())
```

```
dict_values([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```python
[44] # .....................part2 (numerical data)..................
     numeric_data={}
     for sentence in sent_tokens:
         numeric_data[sentence] = 0
         word_tokens = nltk.word_tokenize(sentence)
         for k in word_tokens:
             if k.isdigit():
                 numeric_data[sentence] += 1
     maximum_frequency = max(numeric_data.values())
     for k in numeric_data.keys():
         try:
             numeric_data[k] = (numeric_data[k]/maximum_frequency)
             numeric_data[k] = round(numeric_data[k], 3)
         except ZeroDivisionError:
             x=0
     print(numeric_data.values())
```

```
           print(number_of_data_values())

        dict_values([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```python
[45]  #.....................part3(sentence length)........................
      sent_len_score={}
      for sentence in sent_tokens:
          sent_len_score[sentence] = 0
          word_tokens = nltk.word_tokenize(sentence)
          if len(word_tokens) in range(0,10):
              sent_len_score[sentence]=1-0.05*(10-len(word_tokens))
          elif len(word_tokens) in range(7,20):
              sent_len_score[sentence]=1
          else:
              sent_len_score[sentence]=1-(0.05)*(len(word_tokens)-20)
      for k in sent_len_score.keys():
          sent_len_score[k]=round(sent_len_score[k],4)
      print(sent_len_score.values())
```

```
      dict_values([1, 0.6, 0.0, 1, 0.2, 0.85, -0.2, 0.45, 0.95, 0.9, 1, 1, 0.75, 0.9, 1.0, 1, 0.15, 0.45, 1])
```

```python
[46]  #.....................part4(sentence position)........................
      sentence_position={}
      d=1
      no_of_sent=len(sent_tokens)
      for i in range(no_of_sent):
          a=1/d
          b=1/(no_of_sent-d+1)
          sentence_position[sent_tokens[d-1]]=max(a,b)
          d=d+1
      for k in sentence_position.keys():
          sentence_position[k]=round(sentence_position[k],3)
      print(sentence_position.values())
```

```
      dict_values([1.0, 0.5, 0.333, 0.25, 0.2, 0.167, 0.143, 0.125, 0.111, 0.1, 0.111, 0.125, 0.143, 0.167, 0.2, 0.25, 0.333, 0.5, 1.0])
```

```python
[47]  #Create a frequency table to compute the frequency of each word.
      freqTable = {}
      for word in word_tokens_refined:
          if word in freqTable:
              freqTable[word] += 1
          else:
              freqTable[word] = 1
      for k in freqTable.keys():
          freqTable[k]= math.log10(1+freqTable[k])
      #Compute word frequnecy score of each sentence
      word_frequency={}
      for sentence in sent_tokens:
          word_frequency[sentence]=0
          e=nltk.word_tokenize(sentence)
          f=[]
          for word in e:
              f.append(ps.stem(word))
          for word,freq in freqTable.items():
              if word in f:
                  word_frequency[sentence]+=freq
      maximum=max(word_frequency.values())
      for key in word_frequency.keys():
          try:
              word_frequency[key]=word_frequency[key]/maximum
              word_frequency[key]=round(word_frequency[key],3)
          except ZeroDivisionError:
              x=0
      print(word_frequency.values())
```

```
      dict_values([0.443, 0.443, 0.782, 0.512, 0.767, 0.515, 1.0, 0.62, 0.323, 0.677, 0.373, 0.241, 0.163, 0.237, 0.409, 0.28, 0.708, 0.521, 0.336])
```

```python
[48]  #.....................part 6 (upper cases)................................
      upper_case={}
      for sentence in sent_tokens:
          upper_case[sentence] = 0
          word_tokens = nltk.word_tokenize(sentence)
          for k in word_tokens:
              if k.isupper():
                  upper_case[sentence] += 1
      maximum_frequency = max(upper_case.values())
      for k in upper_case.keys():
          try:
              upper_case[k] = (upper_case[k]/maximum_frequency)
              upper_case[k] = round(upper_case[k], 3)
          except ZeroDivisionError:
              x=0
      print(upper_case.values())
```

```
      dict_values([0.333, 1.0, 0.0, 0.0, 0.333, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.333, 0.667, 0.333, 0.0, 0.0])
```

```python
[49]  #.....................part7 (number of proper noun)..................
      proper_noun={}
      for sentence in sent_tokens:
          tagged_sent = pos_tag(sentence.split())
```

```
        propernouns = [word for word, pos in tagged_sent if pos == 'NNP']
        proper_noun[sentence]=len(propernouns)
    maximum_frequency = max(proper_noun.values())
    for k in proper_noun.keys():
        try:
            proper_noun[k] = (proper_noun[k]/maximum_frequency)
            proper_noun[k] = round(proper_noun[k], 3)
        except ZeroDivisionError:
            x=0
    print(proper_noun.values())
```

dict_values([0.5, 0.167, 0.0, 0.0, 0.833, 0.0, 0.0, 0.0, 0.0, 0.333, 0.167, 0.0, 0.333, 0.0, 0.0, 0.167, 1.0, 0.0, 0.167])

```
[50] #................................ part 8 (word matches with heading) .................
    head_match={}
    heading=sent_tokens[0]
    for sentence in sent_tokens:
        head_match[sentence]=0
        word_tokens = nltk.word_tokenize(sentence)
        for k in word_tokens:
            if k not in stopWords:
                k = ps.stem(k)
                if k in ps.stem(heading):
                    head_match[sentence] += 1
    maximum_frequency = max(head_match.values())
    for k in head_match.keys():
        try:
            head_match[k] = (head_match[k]/maximum_frequency)
            head_match[k] = round(head_match[k], 3)
        except ZeroDivisionError:
            x=0
    print(head_match.values())
```

dict_values([1.0, 0.182, 0.364, 0.182, 0.182, 0.273, 0.455, 0.273, 0.273, 0.182, 0.091, 0.091, 0.091, 0.091, 0.273, 0.091, 0.182, 0.273, 0.091])

```
[51] total_score={}
    for k in cue_phrases.keys():
        total_score[k]=cue_phrases[k]+numeric_data[k]+sent_len_score[k]+sentence_position[k]+word_frequency[k]+upper_case[k]+proper_noun[k]+head_match[k]
    print(total_score.values())
```

dict_values([4.276, 2.892, 1.479, 1.944, 2.515, 1.8050000000000002, 1.398, 1.468, 1.657, 2.192, 1.742, 1.457, 1.48, 1.3949999999999998, 2.215, 2.455, 2.706, 1

```
[52] sumValues = 0
    for sentence in total_score:
        sumValues += total_score[sentence]
    average = sumValues / len(total_score)
    print(average)
    # Storing sentences into our summary.
    summary = ''
    for sentence in sent_tokens:
        if (sentence in total_score) and (total_score[sentence] > (1.2*average)):
            summary += " " + sentence
    print(summary)
```

2.074421052631579
 Coronavirus: How to stay cool in a face mask as hot UK weather forecast. You don't really need the TV weather maps turning red to realise that much of the UK