In [1]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from gensim.models import KeyedVectors
from Functions import calc_similarity
import os
```

In [2]:

```
docs = os.listdir("./Articles")
corpus = []
for i in range(len(docs)):
    with open("./Articles/" + docs[i]) as f:
        corpus.append(f.read())
```

In [3]:

```
print(corpus)
```

```
['I like a bit of pow-wow in any place. Let me rephrase before you think I
am eternally hankering for a fight. What I mean is I would choose crooked
streets over straight highways, sweaty mayhem over pristine elegance. This
is why no matter where I go in this world, coming home to India, and espec
ially Bombay, is never dull blame growing up in the city for my pugilistic
predilection. One of the many descriptors that Mark Twain used in relation
to Bombay was "pow-wow." The place seemed to confound him: "Bewitching".
"Bewildering", "Enchanting", "Arabian Nights come again?" the man was repu
lsed and riveted at the same time. It was a place befitting the number of
exclamations he used."', "Anniversary edition have the feel of a graduatio
n: a year of studious slogging (of which truth be told. my team and I do v
ery little) and madcap fun which we on wish we could indulge in more) roun
ded off with a sense of achievement and lingering anxiety. There's pride t
hat National Geographic Traveller india has lived to see another day and i
n today's precarious media landscape that should account for something. Th
en the gnawing question did we get it right?", 'Our year end edition toast
s ultra indulgence while travelling, featuring itineraries that many will
know to be out of their financial reach. In producing these narratives, I
was struck by a contrast. Travel today is dominated by minimalists or down
sizers, those who preach the gospel of "hard-knock wanderlust." And they a
lmost always reap universal admiration. They are characters to aspire to,
examples of made for-Instagram sayings such as, "All you need is a backpac
k" or "#MotorcycleDiaries." Unable to join these gallivanting philosophers
others marvel at their brave rebellion-oh, to give up the predictability o
f overpriced tourism traps someday, they sigh.', 'For my money, memorable
disagree F ments often centre on food. A friend who was about to settle ab
road was feeling particularly wistful about a storied south Bombay restaur
ant. the kind of eatery that locals like to call "overrated" and guidebook
-toting tourists faithfully make a beeline for. His favourite on the menu?
The baklava-a dry fruit-laden traditional sweet that smacked of decadence
in every bite.']
```

1. Using TF-IDF Vectorization

In [4]:

```
vect = TfidfVectorizer(stop_words="english")
tfidf = vect.fit_transform(corpus)
print(len(vect.vocabulary_))
print(vect.vocabulary_)
```

```
172
{'like': 80, 'bit': 14, 'pow': 111, 'wow': 170, 'place': 110, 'let': 79,
'rephrase': 125, 'think': 147, 'eternally': 43, 'hankering': 64, 'fight':
51, 'mean': 92, 'choose': 21, 'crooked': 27, 'streets': 141, 'straight': 1
40, 'highways': 66, 'sweaty': 144, 'mayhem': 91, 'pristine': 117, 'eleganc
e': 39, 'matter': 90, 'world': 169, 'coming': 24, 'home': 67, 'india': 68,
'especially': 42, 'bombay': 17, 'dull': 36, 'blame': 16, 'growing': 62, 'c
ity': 22, 'pugilistic': 119, 'predilection': 115, 'descriptors': 30, 'mar
k': 88, 'twain': 161, 'used': 165, 'relation': 124, 'confound': 25, 'bewit
ching': 13, 'bewildering': 12, 'enchanting': 40, 'arabian': 6, 'nights': 1
03, 'come': 23, 'man': 87, 'repulsed': 126, 'riveted': 129, 'time': 148,
'befitting': 11, 'number': 104, 'exclamations': 45, 'anniversary': 4, 'edi
tion': 38, 'feel': 49, 'graduation': 61, 'year': 171, 'studious': 143, 'sl
ogging': 135, 'truth': 160, 'told': 151, 'team': 146, 'little': 82, 'madca
p': 85, 'fun': 56, 'wish': 167, 'indulge': 69, 'rounded': 130, 'sense': 13
2, 'achievement': 2, 'lingering': 81, 'anxiety': 5, 'pride': 116, 'nationa
l': 101, 'geographic': 58, 'traveller': 158, 'lived': 83, 'day': 28, 'toda
y': 150, 'precarious': 113, 'media': 93, 'landscape': 78, 'account': 1, 'g
nawing': 59, 'question': 120, 'did': 31, 'right': 128, 'end': 41, 'toast
s': 149, 'ultra': 162, 'indulgence': 70, 'travelling': 159, 'featuring': 4
8, 'itineraries': 72, 'know': 76, 'financial': 52, 'reach': 121, 'producin
g': 118, 'narratives': 100, 'struck': 142, 'contrast': 26, 'travel': 157,
'dominated': 33, 'minimalists': 97, 'downsizers': 34, 'preach': 112, 'gosp
el': 60, 'hard': 65, 'knock': 75, 'wanderlust': 166, 'reap': 122, 'univers
al': 164, 'admiration': 3, 'characters': 20, 'aspire': 7, 'examples': 44,
'instagram': 71, 'sayings': 131, 'need': 102, 'backpack': 8, 'motorcycledi
aries': 99, 'unable': 163, 'join': 73, 'gallivanting': 57, 'philosophers':
109, 'marvel': 89, 'brave': 18, 'rebellion': 123, 'oh': 105, 'predictabili
ty': 114, 'overpriced': 106, 'tourism': 153, 'traps': 156, 'someday': 137,
'sigh': 134, 'money': 98, 'memorable': 94, 'disagree': 32, 'ments': 95, 'c
entre': 19, 'food': 53, 'friend': 54, 'settle': 133, 'abroad': 0, 'feelin
g': 50, 'particularly': 108, 'wistful': 168, 'storied': 139, 'south': 138,
'restaurant': 127, 'kind': 74, 'eatery': 37, 'locals': 84, 'overrated': 10
7, 'guidebook': 63, 'toting': 152, 'tourists': 154, 'faithfully': 46, 'mak
e': 86, 'beeline': 10, 'favourite': 47, 'menu': 96, 'baklava': 9, 'dry': 3
5, 'fruit': 55, 'laden': 77, 'traditional': 155, 'sweet': 145, 'smacked':
136, 'decadence': 29, 'bite': 15}
```

In [5]:

```
similarity = ((tfidf * tfidf.T).A)
print(similarity)
```

```
[[1.         0.01258582 0.         0.03633317]
 [0.01258582 1.         0.0449683  0.        ]
 [0.         0.0449683  1.         0.        ]
 [0.03633317 0.         0.         1.        ]]
```

In [6]:

```
np.fill_diagonal(similarity, np.nan)
```

In [7]:

```
# Taking "Article 1" as the reference/original document for similarity calculation:
# Document which is least similar is:
original_doc_idx = docs.index("article1.txt")
docs[np.nanargmin(similarity[original_doc_idx])]
```

Out[7]:

```
'article3.txt'
```

2. Using Glove Vectors

In [8]:

```
from gensim.test.utils import get_tmpfile
from gensim.models import KeyedVectors
from gensim.scripts.glove2word2vec import glove2word2vec

# Temporary file
tmp_file = get_tmpfile('temp_word2vec.txt')

# GloVe vectors loading function into temporary file
glove2word2vec('Datasets/glove.6B.50d.txt', tmp_file)

# Creating a KeyedVectors from a temporary file
w2v_model = KeyedVectors.load_word2vec_format(tmp_file)
print(type(w2v_model))
```

```
<class 'gensim.models.keyedvectors.Word2VecKeyedVectors'>
```

In [9]:

```
print("Number of words in the vocab: {}".format(len(w2v_model.vocab)))
print("Vector representation of word \"apple\"")
print(w2v_model["apple"])
print("Length of vector for representation of each word")
```

```
Number of words in the vocab: 400000
Vector representation of word "apple"
[ 0.52042  -0.8314    0.49961   1.2893    0.1151    0.057521 -1.3753
 -0.97313   0.18346   0.47672  -0.15112   0.35532   0.25912  -0.77857
  0.52181   0.47695  -1.4251    0.858     0.59821  -1.0903    0.33574
 -0.60891   0.41742   0.21569  -0.07417  -0.5822   -0.4502    0.17253
  0.16448  -0.38413   2.3283   -0.66682  -0.58181   0.74389   0.095015
 -0.47865  -0.84591   0.38704   0.23693  -1.5523    0.64802  -0.16521
 -1.4719   -0.16224   0.79857   0.97391   0.40027  -0.21912  -0.30938
  0.26581 ]
Length of vector for representation of each word
```

In [10]:

```
sim = calc_similarity(w2v_model)
```

In [11]:

```
# Again taking "Article 1" as reference for comparing other documents
source_doc = corpus[0]
target_docs = corpus[1:]

# Calculating similarity scores:
similarity_scores = sim.calculate_similarity(source_doc, target_docs)
```

In [12]:

```
# Just some simple code for better representation
for dic in similarity_scores:
    dic["doc"] = docs[corpus.index(dic["doc"])]
    print(dic)
```

```
{'score': 0.99020684, 'doc': 'article1.txt'}
{'score': 0.98503226, 'doc': 'article3.txt'}
{'score': 0.9721736, 'doc': 'article2.txt'}
```

In [ ]:

```

```