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 thi
nk I am eternally hankering for a fight. What I mean is I would choose
crooked streets over straight highways, sweaty mayhem over pristine el
egance. This is why no matter where I go in this world, coming home to
India, and especially Bombay, is never dull blame growing up in the ci
ty for my pugilistic predilection. One of the many descriptors that Ma
rk Twain used in relation to Bombay was "pow-wow." The place seemed to
confound him: "Bewitching". "Bewildering", "Enchanting", "Arabian Nigh
ts come again?" the man was repulsed and riveted at the same time. It
was a place befitting the number of exclamations he used."', "Annivers
ary edition have the feel of a graduation: a year of studious slogging
(of which truth be told. my team and I do very little) and madcap fun
which we on wish we could indulge in more) rounded off with a sense of
achievement and lingering anxiety. There's pride that National Geograp
hic Traveller india has lived to see another day and in today's precar
ious media landscape that should account for something. Then the gnawi
ng question did we get it right?", 'Our year end edition toasts ultra
indulgence while travelling, featuring itineraries that many will know
to be out of their financial reach. In producing these narratives, I w
as struck by a contrast. Travel today is dominated by minimalists or d
ownsizers, those who preach the gospel of "hard-knock wanderlust." And
they almost always reap universal admiration. They are characters to a
spire to, examples of made for-Instagram sayings such as, "All you nee
d is a backpack" or "#MotorcycleDiaries." Unable to join these galliva
nting philosophers others marvel at their brave rebellion-oh, to give
up the predictability of overpriced tourism traps someday, they sig
h.', 'For my money, memorable disagree F ments often centre on food. A
friend who was about to settle abroad was feeling particularly wistful
about a storied south Bombay restaurant. the kind of eatery that local
s like to call "overrated" and guidebook-toting tourists faithfully ma
ke a beeline for. His favourite on the menu? The baklava-a dry fruit-l
aden 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': 7
9, 'rephrase': 125, 'think': 147, 'eternally': 43, 'hankering': 64, 'f
ight': 51, 'mean': 92, 'choose': 21, 'crooked': 27, 'streets': 141, 's
traight': 140, 'highways': 66, 'sweaty': 144, 'mayhem': 91, 'pristin
e': 117, 'elegance': 39, 'matter': 90, 'world': 169, 'coming': 24, 'ho
me': 67, 'india': 68, 'especially': 42, 'bombay': 17, 'dull': 36, 'bla
me': 16, 'growing': 62, 'city': 22, 'pugilistic': 119, 'predilection':
115, 'descriptors': 30, 'mark': 88, 'twain': 161, 'used': 165, 'relati
on': 124, 'confound': 25, 'bewitching': 13, 'bewildering': 12, 'enchan
ting': 40, 'arabian': 6, 'nights': 103, 'come': 23, 'man': 87, 'repuls
ed': 126, 'riveted': 129, 'time': 148, 'befitting': 11, 'number': 104,
'exclamations': 45, 'anniversary': 4, 'edition': 38, 'feel': 49, 'grad
uation': 61, 'year': 171, 'studious': 143, 'slogging': 135, 'truth': 1
60, 'told': 151, 'team': 146, 'little': 82, 'madcap': 85, 'fun': 56,
'wish': 167, 'indulge': 69, 'rounded': 130, 'sense': 132, 'achievemen
t': 2, 'lingering': 81, 'anxiety': 5, 'pride': 116, 'national': 101,
'geographic': 58, 'traveller': 158, 'lived': 83, 'day': 28, 'today': 1
50, 'precarious': 113, 'media': 93, 'landscape': 78, 'account': 1, 'gn
awing': 59, 'question': 120, 'did': 31, 'right': 128, 'end': 41, 'toas
ts': 149, 'ultra': 162, 'indulgence': 70, 'travelling': 159, 'featurin
g': 48, 'itineraries': 72, 'know': 76, 'financial': 52, 'reach': 121,
'producing': 118, 'narratives': 100, 'struck': 142, 'contrast': 26, 't
ravel': 157, 'dominated': 33, 'minimalists': 97, 'downsizers': 34, 'pr
each': 112, 'gospel': 60, 'hard': 65, 'knock': 75, 'wanderlust': 166,
'reap': 122, 'universal': 164, 'admiration': 3, 'characters': 20, 'asp
ire': 7, 'examples': 44, 'instagram': 71, 'sayings': 131, 'need': 102,
'backpack': 8, 'motorcyclediaries': 99, 'unable': 163, 'join': 73, 'ga
llivanting': 57, 'philosophers': 109, 'marvel': 89, 'brave': 18, 'rebe
llion': 123, 'oh': 105, 'predictability': 114, 'overpriced': 106, 'tou
rism': 153, 'traps': 156, 'someday': 137, 'sigh': 134, 'money': 98, 'm
emorable': 94, 'disagree': 32, 'ments': 95, 'centre': 19, 'food': 53,
'friend': 54, 'settle': 133, 'abroad': 0, 'feeling': 50, 'particularl
y': 108, 'wistful': 168, 'storied': 139, 'south': 138, 'restaurant': 1
27, 'kind': 74, 'eatery': 37, 'locals': 84, 'overrated': 107, 'guidebo
ok': 63, 'toting': 152, 'tourists': 154, 'faithfully': 46, 'make': 86,
'beeline': 10, 'favourite': 47, 'menu': 96, 'baklava': 9, 'dry': 35,
'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 [ ]:

```

```