```
import nltk
nltk.download
nltk.download('punkt')
from nltk.corpus import stopwords

stopwords_en = stopwords.words ("english")

def preprocessing (raw):
   word_list = nltk.word_tokenize (raw)
   text = [w.lower() for w in word_list if w not in stopwords_en]
   return text

f1 = open('test1.txt', 'r')
text1 = preprocessing (f1.read())

f2 = open ('test2.txt', 'r')
text2 = preprocessing (f2.read())
from nltk.probability import FreqDist

f3 = open ('test3.txt', 'r')
text3 = preprocessing (f3.read())
from nltk.probability import FreqDist

word_set = set(text1).union(set(text2)).union(set(text3))


freqd_text1 = FreqDist(text1)
text1_count_dict = dict.fromkeys(word_set,0)
for word in text1:
```

```
    text1_count_dict[word] = freqd_text1[word]

freqd_text2 = FreqDist(text2)
text2_count_dict = dict.fromkeys(word_set,0)
for word in text2:
    text2_count_dict[word]=freqd_text2[word]

freqd_text3 = FreqDist(text3)
text3_count_dict = dict.fromkeys(word_set,0)
for word in text3:
    text3_count_dict[word]=freqd_text3[word]
print(text1_count_dict)
print(text2_count_dict)
print(text3_count_dict)
```

In this code we first download the 'punkt' library and then we read the 3 file and then print the data of three text file.

Step 2

```
freqd_text1 = FreqDist (text1)
text1_length = len(text1)
text1_tf_dict = dict.fromkeys(word_set, 0)
for word in text1:
    text1_tf_dict[word] = freqd_text1[word]/text1_length

freqd_text2 = FreqDist(text2)
text2_length = len(text2)
text2_tf_dict = dict.fromkeys(word_set, 0)
```

```
for word in text2:
   text2_tf_dict[word] = freqd_text2[word]/text2_length


freqd_text3 = FreqDist(text3)
text3_length = len(text3)
text3_tf_dict = dict.fromkeys(word_set, 0)
for word in text3:
   text3_tf_dict[word] = freqd_text3[word]/text3_length
```

in this lines of code we calculate the length of out three files and merge them each other

```
step 3
text12_idf_dict = dict.fromkeys (word_set,0)
text12_length = 3 # 3 documents
for word in text12_idf_dict.keys () :
   if word in text1:

      text12_idf_dict [word] += 1

   if word in text2:
      text12_idf_dict [word] += 1

   if word in text3:
      text12_idf_dict [word] += 1

import math
for word, val in text12_idf_dict.items():
   text12_idf_dict[word] = 1 + math.log(text12_length/ (float(val) ) )
```

import math library and find the values of text files

step 4

```
text1_tfidf_dict = dict.fromkeys (word_set, 0)
for word in text1:
   text1_tfidf_dict[word] = (text1_tf_dict[word]) * (text12_idf_dict[word])

text2_tfidf_dict = dict.fromkeys(word_set, 0)
for word in text2:
   text2_tfidf_dict[word] = (text2_tf_dict[word]) * (text12_idf_dict[word])

text3_tfidf_dict = dict.fromkeys(word_set, 0)
for word in text3:
   text3_tfidf_dict[word] = (text3_tf_dict[word]) * (text12_idf_dict[word])
```

in these line of code we calculating Integrated Development Framework IDF of three text files

step 4

```
import gensim.models.doc2vec
from gensim.models.doc2vec import TaggedDocument

taggeddocs = []
doc1 = TaggedDocument(words=text1, tags=[u'NEWS_1'])
taggeddocs.append(doc1)
doc2 = TaggedDocument(words=text2, tags=[u'NEWS_2'])
taggeddocs.append(doc2)

doc3 = TaggedDocument(words=text3, tags=[u'NEWS_3'])
taggeddocs.append(doc3)

# build the model
```

```
model = gensim.models.Doc2Vec(taggeddocs, dm=0, alpha=0.025, min_alpha=0.025, min_count=0)


# training
for epoch in range (80):
    if epoch % 20 == 0:
        print ('Now training epoch %s' % epoch)


    model.train(taggeddocs,total_examples=model.corpus_count,epochs=model.epochs)
    model.alpha -= 0.002
    model.min_alpha = model.alpha
```

we import the pip install genism in command prompt and train the dataset also decreasing the matching rate and then fix the matching data

step 6

```
v1 = list(text1_tfidf_dict.values())
v2 = list(text2_tfidf_dict.values ())
v3 = list(text3_tfidf_dict.values ())
similarity = 1 - nltk.cluster.cosine_distance(v1,v2)
similarity = 1 - nltk.cluster.cosine_distance(v1,v3)
print ("Similarity Index:{:4.2f} %".format(similarity*100))
```

finally we compare the file which are 3 text files we can also add multiple datasets in out program and calculate the similarity index of files