DOCUMENTATION

tokenisation.py

```
NAME
    tokenisation


FUNCTIONS
    create_tokens_li()
        Function for creating tokens_li and then storing in json file for further usage


    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.


        If the base not specified, returns the natural logarithm (base e) of x.


DATA
    __warningregistry__ = {'version': 10}
    docs = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19...
    freqDist = {}
    i = 146
    snowball_stemmer = <nltk.stem.snowball.SnowballStemmer object>
    tokens_doc = []
    tokens_li = [['time', 'traveller', '(', 'convenient', 'speak', ')', 'e...
    vocabulary = {}
    vocabulary_idf = {}
```

inverted_index.py

```
NAME
   inverted_index


FUNCTIONS
   log(...)
      log(x, [base=math.e])
      Return the logarithm of x to the given base.


      If the base not specified, returns the natural logarithm (base e) of x.


DATA
   count = 2
   dic = {'A': [2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...
   docFiles = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18...
   f = 148
   fname = <_io.TextIOWrapper name='./corpus/148.txt' mode='r' encoding='...
   fp = <_io.TextIOWrapper name='savers/ii.json' mode='w' encoding='cp125...
   i = 148
   item = 'heart'
   l = ['were', 'not', 'so.', 'But', 'to', 'me', 'the', 'future', 'is', '...
   main = ['The', 'Time', 'Traveller', 'convenient', 'speak', 'expounding...
   punc = '!()-[]{;:\'"}\\, <>./?@#$%^&*_~'
   stopwords = <WordListCorpusReader in 'C:\\Users\\chsra\\AppData\\Roami...
   text_tokens = ['were', 'not', 'so', 'But', 'to', 'me', 'the', 'future'...
   tokens_without_sw = ['But', 'future', 'still', 'black', 'blankΓÇ', 'ö'...
   words = 'were not so. But to me the future is still black...al tendern...
```

words.py

```
NAME
   words


FUNCTIONS
   log(...)
       log(x, [base=math.e])
       Return the logarithm of x to the given base.


       If the base not specified, returns the natural logarithm (base e) of x.


   voc_comp()
       Function for retreiving the tokens_li for creating the vocabulary,then storing the vocabulary in a
json file


   voc_construct(document_tokens)
       Function for building the vocabulary i.e. the dictionary which has all the unique words in the
corpus


DATA
   docFiles = ['10.txt', '100.txt', '101.txt', '102.txt', '103.txt', '104...
   freqDist = {}
   snowball_stemmer = <nltk.stem.snowball.SnowballStemmer object>
   tokens_doc = []
   tokens_li = []
   vocabulary = {'!': 452, "'": 2155, '(': 1, ')': 4, ',': 14, '.': 9, '...
   vocabulary_idf = {}
```

tf-idf_values.py

```
NAME
    tf-idf_values


FUNCTIONS
    freq_build(tokens_li)
        function for building the FreqDistribution


    idf_gen()
        function for building the IDF


    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.


        If the base not specified, returns the natural logarithm (base e) of x.


    rIDF(term)
        Function to return corresponding idf
        searching in the vocabulary


    rtf(term, ts, ts_index)
        Function to return the term frequency


DATA
    docs = ['10.txt', '100.txt', '101.txt', '102.txt', '103.txt', '104.txt...
    fp = <_io.TextIOWrapper name='savers/dictionary.json' mode='w' encodin...
    freqDist = {0: FreqDist({'.': 9, ',': 8, 'us': 4, '(': 2, ')': ... 'sa...
    idf = 7.199672344836364

    inner_dict = {0: {1: 0.0, 2: 7.199672344836364, 3: 0.0}, 1: {1: 0.0, 2...

    j = 4848

    json_data = <_io.TextIOWrapper name='savers/words.json' mode='r' encod...

    k = 147

    primaryDictionary = {'!': {0: {1: 0.0, 2: 1.6761103887793516, 3: 0.0},...

    snowball_stemmer = <nltk.stem.snowball.SnowballStemmer object>

    termFreq = 0.03242147769237743

    tokens_doc = []

    tokens_li = [['time', 'traveller', '(', 'convenient', 'speak', ')', 'e...

    ts = ['.', 'future', 'still', 'black', 'blankΓÇ', 'ö', 'vast', 'ignora...

    vocab = 'brittleΓÇ'

    vocabulary = {'!': 452, "'": 2155, '(': 1, ')': 4, ',': 14, '.': 9, '...

    vocabulary_idf = {'!': 46, "'": 1, '(': 8, ')': 8, ',': 146, '.': 146...
```

scoring.py

```
NAME
   scoring

CLASSES
   builtins.object
      main_class

   class main_class(builtins.object)
    |  Methods defined here:
    |
    |  IDF()
    |
    |  freq_gen(tokens_li)
    |
    |  proc_func(query)
    |
    |  rIDF(term)
    |
    |  rtf(term, document_tokens, document_tokens_index)
    |
    |  ter_func()
    |      Function for inputting query and performing query based operations and finally calculating
cosine scores
    |
    |  vocab_gen(document_tokens)
    |
    |  ----------------------------------------------------------------------
    |  Data descriptors defined here:
    |
    |  __dict__
```

```
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  corpusSize = 500
 |
 |  docs = ['10.txt', '100.txt', '101.txt', '102.txt', '103.txt', '104.txt...
 |
 |  freqDist = {}
 |
 |  queryStr = ''
 |
 |  snowball_stemmer = <nltk.stem.snowball.SnowballStemmer object>
 |
 |  tokens_doc = []
 |
 |  tokens_li = []
 |
 |  vocabulary = {}
 |
 |  vocabulary_idf = {}

FUNCTIONS
    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.
        If the base not specified, returns the natural logarithm (base e) of x.
```

trail.py

```
NAME
    trail

FUNCTIONS
    log(...)
        log(x, [base=math.e])
        Return the logarithm of x to the given base.

        If the base not specified, returns the natural logarithm (base e) of x.

DATA
    __warningregistry__ = {'version': 14}
    i = "'Page: 17'"
    query = 'time traveller'
    result = ["'Page: 23'", "'Page: 8'", "'Page: 146'", "'Page: 11'", "'Pa...
```