Lab Course - Distributed Data Analytics
Exercise 2

1.1 Data Cleaning and Text Tokenisation:

Function name : read_data
Parameter : Directory location

This function reads the extracts the documents from the directory, reads the text from the document and returns the list of the text per document.

```
def read_data(directory):
    if print_time: start = MPI.Wtime()
    filelist = []
    text_doc = []
    filename = []
    for root,dirs,files in os.walk(directory):
        files = [os.path.join(root, f) for f in files]
        for file in files:
            filename.append(os.path.basename(file))
            f = open(file,'rb')
            text_doc.append(''.join(map(str, f.readlines())))
    if print_time: print("Rank: ",rank,"Reading Data: ",
                         round(MPI.Wtime() - start,4))
    return text_doc
```

Function name : clean_token
Parameter : List of text per document

This function checks for the stopwords (provided by NLTK library) within the text and removes them. It uses word_tokenize of NLTK to tokenize the words per document and removes the words that are not alphabetic, changes the case of the words to lower and returns a list of tokens per document

```
def clean_token(data):
    if print_time: start = MPI.Wtime()
    stopword_set = set(stopwords.words('english'))
    text_data = []
    for i,text in enumerate(data):
        text = word_tokenize(text)
        text = [word for word in text if word.isalpha()]
        text = [t.lower() for t in text]
        text = [word for word in text if word not in stopword_set]
        text_data.append(text)
    if print_time: print("Rank: ",rank,"Cleaning Data: ",
                         round(MPI.Wtime() - start,4))
    return text_data
```

Function name : tf
Parameter : list of tokens per document

This function uses the list of tokens per document and calculates the Term Frequency (TF) by counting the number of tokes a token occurs in the document and returns the same.

$$\mathrm{TF}(t,d) = \frac{n^d(t)}{\sum_{t' \in d} n^d(t')}$$

```
def tf(data):
    if print_time: start = MPI.Wtime()
    count = [dict(Counter(token)) for token in data]
    tf_token = [dict(Counter(token)) for token in data]

    for doc in tf_token:
        token_num = len(doc)
        for token in doc:
            doc[token] = doc[token] / token_num
    if print_time: print("Rank: ",rank,"TF Calculation: "
                         round(MPI.Wtime() - start,4))
    return tf_token,count
```

Function name : token_doc
Parameter : list of tokens per document

This function uses the list of tokens per document and returns the number of documents in which a particular token appears. This is used to calculate Inverse Document Frequency (IDF)

```
def token_doc(tf_doc):
    tokdoc_freq = {}
    for tf_token in tf_doc:
        for token in tf_token:
            if token not in tokdoc_freq.keys():
                tokdoc_freq[token] = 1
            else:
                tokdoc_freq[token] = tokdoc_freq[token] + 1
    return tokdoc_freq
```

Function name : idf
Parameter : List of tokens in corpus and Token frequency in corpus

This function uses the list of tokens and its frequency in the corpus to calculate the Inverse Document Frequency of the token. IDF is counting the number of documents in the corpus and counting the number of documents that contain a token

$$\mathrm{IDF}(t) = \log \frac{|C|}{\sum_{d \in C} \mathbb{I}(t,d)}$$

```
def idf(tok_list,token_doccnt):
    if print_time: start = MPI.Wtime()
    idf_doccnt = {}
    for token in tok_list:
        idf_doccnt[token] = np.log(doc_num/token_doccnt[token])
    if print_time: print("Rank: ",rank,"IDF Calculation: ",
                         round(MPI.Wtime() - start,4))
    return idf_doccnt
```

Function name : tfidf
Parameter : TF and IDF of the token

This function calculates the Term Frequency – Inverse Document Frequency (TF-IDF) of a token using the previously calculated TF and IDF of the token.

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t)$$

```
def tfidf(tokcnt,idf_token):
    if print_time: start = MPI.Wtime()
    tfidf_token = tokcnt
    for doc in tfidf_token:
        for token in doc:
            doc[token] = doc[token] * idf_token[token]
    if print_time: print("Rank: ",rank,"TF-IDF Calculation: ",
                         round(MPI.Wtime() - start,4))
    return tfidf_token
```

MPI Parallelisation Flow:

```
if rank == 0:
    directory = "G:/DA - Hildeshim/DDA Lab/Exercise 2/Dataset/"
    data = read_data(directory)
    data_workers = np.array_split(data,size-1)
    for workers in range(1,size):
        comm.send(data_workers[workers-1],dest = workers,tag=1)
        doc_token = comm.recv(source = workers,tag=10)
        token.extend(doc_token)
        comm.send(doc_token,dest = workers,tag=2)
        tf_tok= comm.recv(source = workers,tag=20)
        tf_token.extend(tf_tok[0])
        token_count.extend(tf_tok[1])
    token_doccnt = token_doc(token_count)
    tokdoc_workers = np.array_split(list(token_doccnt.keys()),size-1)
    for workers in range(1,size):
        comm.send(tokdoc_workers[workers-1],dest = workers,tag=3)
        comm.send(token_doccnt,dest = workers,tag = 4)
        idf_tok = comm.recv(source = workers, tag=30)
        idf_token.update(idf_tok)
    token_count_workers = np.array_split(token_count,size-1)
    for workers in range(1,size):
        comm.send(token_count_workers[workers-1],dest = workers,tag=5)
        comm.send(idf_token,dest = workers,tag=6)
        tfidf_tok = comm.recv(source = workers, tag=40)
        tfidf_token.extend(tfidf_tok)
else:
    data_worker = comm.recv(source=0,tag=1)
    comm.send(clean_token(data_worker), dest = 0,tag=10)
    tf_worker = comm.recv(source=0,tag=2)
    comm.send(tf(tf_worker),dest=0,tag=20)
    tokdoc_worker= comm.recv(source = 0,tag=3)
    token_doccnt = comm.recv(source = 0,tag = 4)
    comm.send(idf(tokdoc_worker,token_doccnt),dest = 0,tag=30)
    tokcnt_worker = comm.recv(source = 0,tag = 5)
    idf_token = comm.recv(source = 0,tag = 6)
    comm.send(tfidf(tokcnt_worker,idf_token),dest = 0,tag=40)
```

Reading the text:

In this lab exercise, I've considered the process with rank 0 as Master and other processes as workers. First of all the master reads the text from documents and outputs a list of text per document in corpus. The master then splits the data based on the number of processes and sends them to the worker to perform data cleaning, tokenization and term frequency calculation.

Process	Reading the Text
2	0.1476
4	0.1401
6	0.1540

Data: ['b\'From: strom@Watson.Ibm.Com (Rob Strom)\\n\'b\'Subject: Re: [soc.motss, et al.] "Princeton axes matching funds for B oy Scouts"\\n\'b\'Distribution: usa\\n\'b\'Organization: IBM Research\\n\'b\'Lines: 15\\n\'b\'\\n\'b\'In article <N4HY.93Apr512 0934@harder.ccr-p.ida.org>, n4hy@harder.ccr-p.ida.org (Bob McGwier) writes:\\n\'b\'\\n\'b\'|> [1] HOWEVER, I hate economic terr orism and political correctness\\n\'b\'|> worse than I hate this policy. \\n\'b\'\\n\'b\'\\n\'b\'|> [2] A more effective appro ach is to stop donating\\n\'b\'|> to ANY organizating that directly or indirectly supports gay rights issues\\n\'b\'|> until th ey end the boycott on funding of scouts. \\n\'b\'\\n\'b\'Can somebody reconcile the apparent contradiction between [1] and [2]?\\n\'b\'\\n\'b\'-- \\n\'b\'Rob Strom, strom@watson.ibm.com, (914) 784-7641\\n\'b\'IBM Research, 30 Saw Mill River Road, P. O. Box 704, Yorktown Heights, NY 10598\\n\'', 'b\'From: keith@cco.caltech.edu (Keith Allan Schneider)\\n\'b\'Subject: Re: Poli tical Atheists?\\n\'b\'Organization: California Institute of Technology, Pasadena\\n\'b\'Lines: 11\\n\'b\'NNTP-Posting-Host: pu nisher.caltech.edu\\n\'b\'\\n\'b\'arromdee@jyusenkyou.cs.jhu.edu (Ken Arromdee) writes:\\n\'b\'\\n\'b\'>>The motto originated i n the Star-Spangled Banner. Tell me that this has\\n\'b\'>>something to do with atheists.\\n\'b\'>The motto _on_coins_ origina ted as a McCarthyite smear which equated atheism\\n\'b\'>with Communism and called both unamerican.\\n\'b\'\\n\'b"No it didn \'t. The motto has been on various coins since the Civil War.\\n"b"It was just required to be on *all* currency in the 50 \'s.\\n"b\'\\n\'b\'keith\\n\'', 'b\'From: keith@cco.caltech.edu (Keith Allan Schneider)\\n\'b\'Subject: Re: >>>>>>Pompous ass \\n\'b\'Organization: California Institute of Technology, Pasadena\\n\'b\'Lines: 9\\n\'b\'NNTP-Posting-Host: punisher.caltech.e du\\n\'b\'\\n\'b\'kmr4@po.CWRU.edu (Keith M. Ryan) writes:\\n\'b\'\\n\'b\'>>Then why do people keep asking the same questions o ver and over?\\n\'b\'>Because you rarely ever answer them.\\n\'b\'\\n"b"Nope, I\'ve answered each question posed, and most wer e answered multiple\\n"b\'times.\\n\'b\'\\n\'b\'keith\\n\'', 'b\'From: keith@cco.caltech.edu (Keith Allan Schneider)\\n\'b\'Sub ject: Re: >>>>>>Pompous ass\\n\'b\'Organization: California Institute of Technology, Pasadena\\n\'b\'Lines: 14\\n\'b\'NNTP-Post ing-Host: punisher.caltech.edu\\n\'b\'\\n\'b\'livesey@solntze.wpd.sgi.com (Jon Livesey) writes:\\n\'b\'\\n\'b\'>>>How long does it [the motto] have to stay around before it becomes the\\n\'b"b">>>default? ... Where\'s the cutoff point? \\n"b">>I don\'t kn ow where the exact cutoff is, but it is at least after a few\\n"b\'>>years, and surely after 40 years.\\n\'b\'>Why does the not ion of default not take into account changes\\n\'b\'>in population makeup? \\n\'b\'\\n\'b\'Specifically, which changes are you talking about? Are you arguing\\n\'b\'that the motto is interpreted as offensive by a larger portion of the\\n\'b\'populat ion now than 40 years ago?\\n\'b\'\\n\'b\'keith\\n\'']

Exercise 1: Data Cleaning and Text Tokenization:

In order to preprocess the data, the split list of text per document is sent to all the workers by the master. The workers perform the data cleaning by removing the punctuations, numbers and the list of common English stopwords that are not useful for building the model. Also, it tokenizes the documents and sends it back to the master. The master receives the list of tokens per document in corpus and uses it as input for calculating TF

Process	Cleaning and Tokenization
2	7.4077
4	8.4247
6	8.0889

Token: [['strom', 'rob', 'strom', 'et', 'al', 'princeton', 'axes', 'matching', 'funds', 'boy', 'scouts', 'ibm', 'article', 'bo b', 'mcgwier', 'writes', 'however', 'hate', 'economic', 'terrorism', 'political', 'worse', 'hate', 'policy', 'effective', 'appr oach', 'stop', 'organizating', 'directly', 'indirectly', 'supports', 'gay', 'rights', 'end', 'boycott', 'funding', 'scouts', 's omebody', 'reconcile', 'apparent', 'contradiction', 'strom', 'strom', 'research', 'saw', 'mill', 'river', 'road', 'box', 'yorkt own', 'heights', 'ny'], ['keith', 'keith', 'allan', 'schneider', 'political', 'atheists', 'california', 'institute', 'technolog y', 'ken', 'arromdee', 'writes', 'motto', 'originated', 'banner', 'tell', 'something', 'motto', 'originated', 'mccarthyite', 's mear', 'equated', 'communism', 'called', 'motto', 'various', 'coins', 'since', 'civil', 'b', 'required', 'currency'], ['keith', 'keith', 'allan', 'schneider', 'pompous', 'california', 'institute', 'technology', 'keith', 'ryan', 'writes', 'people', 'keep', 'asking', 'questions', 'rarely', 'ever', 'answer', 'nope', 'answered', 'question', 'posed', 'answered'], ['keith', 'keith', 'al lan', 'schneider', 'pompous', 'california', 'institute', 'technology', 'jon', 'livesey', 'writes', 'long', 'motto', 'stay', 'ar ound', 'becomes', 'default', 'cutoff', 'point', 'b', 'know', 'exact', 'cutoff', 'least', 'b', 'years', 'surely', 'notion', 'def ault', 'take', 'account', 'population', 'makeup', 'changes', 'talking', 'motto', 'interpreted', 'offensive', 'larger', 'portio n', 'years', 'ago']]

Exercise 2: Calculate Term Frequency (TF):

The cleaned and tokenized text received by the master is sent again to the workers for calcualting TF. Once the workers receive the required data, they calculate the TF and sends it back to master. The master then groups the results provided by the workers to a single list of frequency of tokens in corpus.

Process	TF
2	0.1415
4	0.1715
6	0.1235


```
TF: [{'writes': 1, 'boycott': 1, 'ny': 1, 'strom': 4, 'matching': 1, 'hate': 2, 'worse': 1, 'research': 1, 'river': 1, 'stop':
1, 'road': 1, 'organizating': 1, 'rob': 1, 'mill': 1, 'saw': 1, 'ibm': 1, 'al': 1, 'directly': 1, 'boy': 1, 'effective': 1, 'ar
ticle': 1, 'political': 1, 'gay': 1, 'scouts': 2, 'bob': 1, 'reconcile': 1, 'axes': 1, 'apparent': 1, 'however': 1, 'box': 1,
'contradiction': 1, 'et': 1, 'supports': 1, 'funds': 1, 'approach': 1, 'mcgwier': 1, 'indirectly': 1, 'funding': 1, 'princeto
n': 1, 'rights': 1, 'economic': 1, 'heights': 1, 'yorktown': 1, 'policy': 1, 'end': 1, 'terrorism': 1, 'somebody': 1}, {'requir
ed': 1, 'since': 1, 'political': 1, 'arromdee': 1, 'b': 1, 'currency': 1, 'something': 1, 'schneider': 1, 'technology': 1, 'kei
th': 2, 'various': 1, 'ken': 1, 'allan': 1, 'called': 1, 'california': 1, 'civil': 1, 'atheists': 1, 'communism': 1, 'writes':
1, 'banner': 1, 'institute': 1, 'originated': 2, 'motto': 3, 'smear': 1, 'equated': 1, 'mccarthyite': 1, 'tell': 1, 'coins':
1}, {'technology': 1, 'answer': 1, 'keep': 1, 'allan': 1, 'writes': 1, 'question': 1, 'california': 1, 'questions': 1, 'pompou
s': 1, 'people': 1, 'asking': 1, 'schneider': 1, 'institute': 1, 'rarely': 1, 'ever': 1, 'keith': 3, 'nope': 1, 'posed': 1, 'ry
an': 1, 'answered': 2}, {'livesey': 1, 'surely': 1, 'exact': 1, 'writes': 1, 'around': 1, 'take': 1, 'makeup': 1, 'jon': 1, 'po
int': 1, 'larger': 1, 'schneider': 1, 'years': 2, 'ago': 1, 'portion': 1, 'know': 1, 'keith': 2, 'b': 2, 'population': 1, 'offe
nsive': 1, 'long': 1, 'allan': 1, 'changes': 1, 'california': 1, 'cutoff': 2, 'stay': 1, 'interpreted': 1, 'pompous': 1, 'becom
es': 1, 'talking': 1, 'least': 1, 'motto': 2, 'institute': 1, 'account': 1, 'notion': 1, 'technology': 1, 'default': 2}]
```


Exercise 3: Calculate Inverse Document Frequency (IDF):

Once the master groups the TF, it splits the list of tokens in corpus into groups and sends them to the worker. Also it sends the entire copy of number of documents in which a token occurs. This is received by the workers and IDF is calculated which is sent back to the master. The master receives the data from workers and groups them further to be used for calculating TF-IDF

Process	IDF
2	0.0618
4	0.0604
6	0.0674


```
IDF {'keep': 1.3862943611198906, 'writes': 0.0, 'something': 1.3862943611198906, 'point': 1.3862943611198906, 'end': 1.38629436
11198906, 'know': 1.3862943611198906, 'strom': 1.3862943611198906, 'matching': 1.3862943611198906, 'keith': 0.2876820724517808
5, 'hate': 1.3862943611198906, 'worse': 1.3862943611198906, 'exact': 1.3862943611198906, 'research': 1.3862943611198906, 'offen
sive': 1.3862943611198906, 'portion': 1.3862943611198906, 'road': 1.3862943611198906, 'stay': 1.3862943611198906, 'nope': 1.386
2943611198906, 'communism': 1.3862943611198906, 'organizating': 1.3862943611198906, 'least': 1.3862943611198906, 'saw': 1.38629
43611198906, 'people': 1.3862943611198906, 'equated': 1.3862943611198906, 'ibm': 1.3862943611198906, 'directly': 1.386294361119
8906, 'boy': 1.3862943611198906, 'institute': 0.28768207245178085, 'required': 1.3862943611198906, 'civil': 1.3862943611198906,
'political': 0.6931471805599453, 'apparent': 1.3862943611198906, 'scouts': 1.3862943611198906, 'arromdee': 1.3862943611198906,
'b': 0.6931471805599453, 'makeup': 1.3862943611198906, 'smear': 1.3862943611198906, 'box': 1.3862943611198906, 'long': 1.386294
3611198906, 'stop': 1.3862943611198906, 'default': 1.3862943611198906, 'article': 1.3862943611198906, 'various': 1.386294361119
8906, 'funding': 1.3862943611198906, 'changes': 1.3862943611198906, 'allan': 0.28768207245178085, 'atheists': 1.386294361119890
6, 'approach': 1.3862943611198906, 'california': 0.28768207245178085, 'banner': 1.3862943611198906, 'questions': 1.386294361119
8906, 'pompous': 0.6931471805599453, 'mccarthyite': 1.3862943611198906, 'terrorism': 1.3862943611198906, 'coins': 1.38629436111
98906, 'livesey': 1.3862943611198906, 'since': 1.3862943611198906, 'larger': 1.3862943611198906, 'answered': 1.386294361119890
6, 'ryan': 1.3862943611198906, 'currency': 1.3862943611198906, 'boycott': 1.3862943611198906, 'ny': 1.3862943611198906, 'jon':
1.3862943611198906, 'population': 1.3862943611198906, 'talking': 1.3862943611198906, 'schneider': 0.28768207245178085, 'rarel
y': 1.3862943611198906, 'ever': 1.3862943611198906, 'becomes': 1.3862943611198906, 'bob': 1.3862943611198906, 'called': 1.38629
43611198906, 'cutoff': 1.3862943611198906, 'posed': 1.3862943611198906, 'asking': 1.3862943611198906, 'rob': 1.386294361119890
6, 'mill': 1.3862943611198906, 'effective': 1.3862943611198906, 'motto': 0.6931471805599453, 'ken': 1.3862943611198906, 'al':
1.3862943611198906, 'question': 1.3862943611198906, 'account': 1.3862943611198906, 'surely': 1.3862943611198906, 'gay': 1.38629
43611198906, 'river': 1.3862943611198906, 'reconcile': 1.3862943611198906, 'take': 1.3862943611198906, 'axes': 1.38629436111989
06, 'around': 1.3862943611198906, 'years': 1.3862943611198906, 'however': 1.3862943611198906, 'contradiction': 1.38629436111989
06, 'supports': 1.3862943611198906, 'funds': 1.3862943611198906, 'economic': 1.3862943611198906, 'ago': 1.3862943611198906, 'mc
gwier': 1.3862943611198906, 'indirectly': 1.3862943611198906, 'answer': 1.3862943611198906, 'et': 1.3862943611198906, 'princeto
n': 1.3862943611198906, 'rights': 1.3862943611198906, 'interpreted': 1.3862943611198906, 'notion': 1.3862943611198906, 'tell':
1.3862943611198906, 'originated': 1.3862943611198906, 'heights': 1.3862943611198906, 'yorktown': 1.3862943611198906, 'policy':
1.3862943611198906, 'somebody': 1.3862943611198906, 'technology': 0.28768207245178085}
```

Exercise 4: Calculate Term Frequency - Inverse Document Frequency (TF-IDF):

Once the master groups the IDF, it splits the term frequency of token per document in corpus and sends them to workers. Also, a copy of the IDF to all the workers. The workers perform TF-IDF calcualtion and send the result back to master which is further grouped up.

Process	IDF
2	0.0778
4	0.1125
6	0.1127


```
TFIDF [{'writes': 0.0, 'boycott': 1.3862943611198906, 'ny': 1.3862943611198906, 'strom': 5.545177444479562, 'matching': 1.38629
43611198906, 'hate': 2.772588722239781, 'worse': 1.3862943611198906, 'research': 1.3862943611198906, 'river': 1.386294361119890
6, 'stop': 1.3862943611198906, 'road': 1.3862943611198906, 'organizating': 1.3862943611198906, 'rob': 1.3862943611198906, 'mil
l': 1.3862943611198906, 'saw': 1.3862943611198906, 'ibm': 1.3862943611198906, 'al': 1.3862943611198906, 'directly': 1.386294361
1198906, 'boy': 1.3862943611198906, 'effective': 1.3862943611198906, 'article': 1.3862943611198906, 'political': 0.693147180559
9453, 'gay': 1.3862943611198906, 'scouts': 2.772588722239781, 'bob': 1.3862943611198906, 'reconcile': 1.3862943611198906, 'axe
s': 1.3862943611198906, 'apparent': 1.3862943611198906, 'however': 1.3862943611198906, 'box': 1.3862943611198906, 'contradictio
n': 1.3862943611198906, 'et': 1.3862943611198906, 'supports': 1.3862943611198906, 'funds': 1.3862943611198906, 'approach': 1.38
62943611198906, 'mcgwier': 1.3862943611198906, 'indirectly': 1.3862943611198906, 'funding': 1.3862943611198906, 'princeton': 1.
3862943611198906, 'rights': 1.3862943611198906, 'economic': 1.3862943611198906, 'heights': 1.3862943611198906, 'yorktown': 1.38
62943611198906, 'policy': 1.3862943611198906, 'end': 1.3862943611198906, 'terrorism': 1.3862943611198906, 'somebody': 1.3862943
611198906}, {'required': 1.3862943611198906, 'since': 1.3862943611198906, 'political': 0.6931471805599453, 'arromdee': 1.386294
3611198906, 'b': 0.6931471805599453, 'writes': 0.0, 'something': 1.3862943611198906, 'schneider': 0.28768207245178085, 'technol
ogy': 0.28768207245178085, 'keith': 0.5753641449035617, 'various': 1.3862943611198906, 'ken': 1.3862943611198906, 'allan': 0.28
768207245178085, 'called': 1.3862943611198906, 'california': 0.28768207245178085, 'civil': 1.3862943611198906, 'atheists': 1.38
62943611198906, 'communism': 1.3862943611198906, 'currency': 1.3862943611198906, 'banner': 1.3862943611198906, 'institute': 0.2
8768207245178085, 'originated': 2.772588722239781, 'motto': 2.0794415416798357, 'smear': 1.3862943611198906, 'equated': 1.38629
43611198906, 'coins': 1.3862943611198906, 'tell': 1.3862943611198906, 'mccarthyite': 1.3862943611198906}, {'technology': 0.2876
8207245178085, 'answer': 1.3862943611198906, 'answered': 2.772588722239781, 'allan': 0.28768207245178085, 'writes': 0.0, 'quest
ion': 1.3862943611198906, 'california': 0.28768207245178085, 'questions': 1.3862943611198906, 'posed': 1.3862943611198906, 'eve
r': 1.3862943611198906, 'asking': 1.3862943611198906, 'schneider': 0.28768207245178085, 'institute': 0.28768207245178085, 'rare
ly': 1.3862943611198906, 'people': 1.3862943611198906, 'pompous': 0.6931471805599453, 'keith': 0.8630462173553426, 'nope': 1.38
62943611198906, 'ryan': 1.3862943611198906, 'keep': 1.3862943611198906}, {'livesey': 1.3862943611198906, 'surely': 1.3862943611
198906, 'ago': 1.3862943611198906, 'b': 1.3862943611198906, 'writes': 0.0, 'makeup': 1.3862943611198906, 'take': 1.386294361119
8906, 'around': 1.3862943611198906, 'default': 2.772588722239781, 'jon': 1.3862943611198906, 'years': 2.772588722239781, 'talki
ng': 1.3862943611198906, 'schneider': 0.28768207245178085, 'know': 1.3862943611198906, 'point': 1.3862943611198906, 'portion':
1.3862943611198906, 'keith': 0.5753641449035617, 'exact': 1.3862943611198906, 'population': 1.3862943611198906, 'offensive': 1.
3862943611198906, 'long': 1.3862943611198906, 'allan': 0.28768207245178085, 'changes': 1.3862943611198906, 'california': 0.2876
8207245178085, 'cutoff': 2.772588722239781, 'stay': 1.3862943611198906, 'interpreted': 1.3862943611198906, 'pompous': 0.6931471
805599453, 'becomes': 1.3862943611198906, 'larger': 1.3862943611198906, 'least': 1.3862943611198906, 'motto': 1.386294361119890
6, 'account': 1.3862943611198906, 'notion': 1.3862943611198906, 'technology': 0.28768207245178085, 'institute': 0.2876820724517
8085}]
```


Example:

Token	Strom
TF	4
IDF	1.386
TF-IDF	5.545 (1.386 x 4)