DDA LAB

SIMRAN KAUR

311443

Exercise Sheet 2 ¶

Complex Data Lab: Processing Text Data in a Distributed Setting

In this exercise sheet, you are going to apply distributed computing concepts using Message Passing Interface API provided by Python (mpi4py) to Natural Language Processing (NLP) techniques using complex data in the form of text documents. The NLP application uses machine learning models to understand the human language and speech. The text data is usually large and consists of complex structures. In this lab you will use MPI framework to process large natural language corpora. More precisely, you are going to do some basic tasks in NLP including data cleaning, text tokenization and convert words into their Term Frequency, Inverse Document Frequency (TFIDF) scores.

Dataset and a scikit-learn

You will use "Twenty Newsgroups" corpus (data set) available at UCI repository https://archive (https://archive). ics.uci.edu/ml/datasets/Twenty+Newsgroups. It consists of 20 subfolders each belong to a specify topic. Each subfolder has multiple documents.

You can look at the the following blog post to get yourself familiarized with TFIDF calculation available in scikit-learn (https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a (https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a)). Note: This is just a tutorial you will not use scikit-learn to solve the final task.

Exercise 1: Data cleaning and text tokenization

In a text document you encounter words that are not helpful for your final model. For example, punctuations and numbers, meaningless words, common English stopwords etc. Your first task is to preprocess your data so you can remove these words from the documents. Your solution should be based on MPI framework. You have to develop (write code) a distributed algorithm that cleans the data by removing the above mentioned words/numbers. You can take help of

the data by removing the above mentioned words/numbers. You can take help of some python libraries i.e. NLTK. The developed program should take a set of documents in raw format as input, and outputs a set of documents that are cleaned and tokenized.

Please explain your solution and how you distribute work among workers.

1. Cleaning: remove all punctuations, numbers and The list of common English stopwords used in this exercise sheet can be found in the reference [4].

2. Tokenize: Tokenize your documents so it is easy to process in the next task i.e. Tokenize words and output as a comma separated document.

In [2]:

```
import pandas as pd
```

The folders in the twenty_newsgroup are first divided among workers, depending on the number of workers provided, using scatter which divides equal data among all workers and then each worker cleans and tokenize it's part and returns a list of lists containing words from each document. In the end at the master node all of the files from each worker are stored in a list whose each entry is a list corresponding to each document. This is then stored in a text file with each row having words from each document.

In [ ]:

```
# Importing Libraries
import os
import numpy as np
import nltk
from nltk.corpus import stopwords
import re
from mpi4py import MPI

# Setting MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
set_files = None

startTime = MPI.Wtime()
# Master node
if rank == 0:
    main = "C:/DDA/20_newsgroups"    # main path of the folder
    directory = os.scandir(main)
    list_files = []      # list stores the path of the subfolders
    for i in directory:
        newpath = os.path.join(i)
        newpath2 = main + '/' + newpath.split("\\")[-1]
        list_files.append(newpath2)

    num_files = len(list_files)
    step = num_files//size

    set_files = []     # stores chunks of paths for each worker

    for i in range(size - 1):
        set_files.append(list_files[i*step:(i+1)*step])
    set_files.append(list_files[(size - 1)*step: ])

worker_chunk = comm.scatter(set_files, root = 0)

# cleaning and tokenizing data
clean_data = []
for path in worker_chunk:
    sub_directory = os.scandir(path)
    for j in sub_directory:                 # text files in each folder provided to workers
        files = []
        filepath = os.path.join(j)
        newpath3 = path + '/' + filepath.split("\\")[-1]
        f = open(newpath3, "r")
        file = f.read()
        file = re.sub(r'[^\w\s]', '', file)        # removing punctuations
        file = re.sub(r'[_]+', '', file)          # removing _ as it is part of \w
        file = re.sub(r'[\d]+', '', file)           # removing digits
        file= re.sub(r'\b\w{1}\b', '', file)        # removing words consisting of single c
        file = re.sub(r'\b\w{15,}\b', '', file)       # removing words of length 15 or more
        tokens = nltk.word_tokenize(file)           # tokenizing data
        stop_words = stopwords.words('english')        # list of stopwords
        for word in tokens:
            if word.lower() not in stop_words:        # appending words other than stopword
                files.append(word.lower())
        clean_data.append(files)

processed_data = comm.gather(clean_data, root = 0)      # gathering cleaned and tokenized d
```

```
# Each worker returns list of lists where inner list consists of tokens from each text file

if rank == 0:
    tokenized_data = [item for sublist in processed_data for item in sublist]#converting li
    with open('tokenized_data.txt', 'a') as f:    # saving cleaned data into text file to u
        for doc in tokenized_data:
            for line in doc:
                f.write(line)
                f.write(' ')             # individual words in text file are seperated usin
            f.write('\n')                  # new document starts from new line
    print('Time:', MPI.Wtime()-startTime)
```

Results of tokenizing when used 2, 4 and 8 processors respectively.

Processors : 2

```
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq atheist resources summary books addresses music a
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq introduction atheism summary please read file pos
newsgroups altatheism path idbsturztubsde benedikt rosenau subject gospel dating messageid sender mr nntp inews entry organization tec
xref altatheism path mathew newsgroups subject university violating separation churchstate messageid date mon apr gmt references aakep
xref altatheism socmotss recscouting newsgroups path rob strom subject socmotss et al princeton axes matching funds boy scouts sender
newsgroups altatheism path idbsturztubsde benedikt rosenau subject visit jehovahs witnesses messageid sender mr nntp inews entry organ
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references organization california
newsgroups altatheism path idbsturztubsde benedikt rosenau subject anecdote islam messageid sender mr nntp inews entry organization te
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu marbmerhbnrca km
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu organization cal
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
xref altatheism talkorigins path keith allan schneider newsgroups subject albert sabin messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path newsgroups altatheism subject dont innocents die without death penalty messageid viceicotekcom robert beauchaine date apr gmt ref
path newsgroups altatheism subject ancient islamic rituals messageid viceicotekcom robert beauchaine date apr gmt references organizat
path newsgroups altatheism subject political atheists messageid viceicotekcom robert beauchaine date apr gmt references organization t
xref altatheism talkorigins newsgroups path scott mullins subject return abused creationist thread real probability abiogenesis messag
xref altatheism path newsgroups subject list biblical contradictions messageid bluecispittedu david joslin date apr gmt sender newspit
path newsgroups altatheism subject must creator maybe messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbea
path newsgroups altatheism subject americans evolution messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbe
path newsgroups altatheism subject speculations messageid aprbmerhbnrca douglas graham date apr gmt sender newsbmerhbnrca usenet news
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
```

Processors : 4

```
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq atheist resources summary books addresses music a
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq introduction atheism summary please read file pos
newsgroups altatheism path idbsturztubsde benedikt rosenau subject gospel dating messageid sender mr nntp inews entry organization tec
xref altatheism path mathew newsgroups subject university violating separation churchstate messageid date mon apr gmt references aakep
xref altatheism socmotss recscouting newsgroups path rob strom subject socmotss et al princeton axes matching funds boy scouts sender
newsgroups altatheism path idbsturztubsde benedikt rosenau subject visit jehovahs witnesses messageid sender mr nntp inews entry organ
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references organization california
newsgroups altatheism path idbsturztubsde benedikt rosenau subject anecdote islam messageid sender mr nntp inews entry organization te
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu marbmerhbnrca km
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu organization cal
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
xref altatheism talkorigins path keith allan schneider newsgroups subject albert sabin messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path newsgroups altatheism subject dont innocents die without death penalty messageid viceicotekcom robert beauchaine date apr gmt ref
path newsgroups altatheism subject ancient islamic rituals messageid viceicotekcom robert beauchaine date apr gmt references organizat
path newsgroups altatheism subject political atheists messageid viceicotekcom robert beauchaine date apr gmt references organization t
xref altatheism talkorigins newsgroups path scott mullins subject return abused creationist thread real probability abiogenesis messag
xref altatheism path newsgroups subject list biblical contradictions messageid bluecispittedu david joslin date apr gmt sender newspit
path newsgroups altatheism subject must creator maybe messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbea
path newsgroups altatheism subject americans evolution messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbe
path newsgroups altatheism subject speculations messageid aprbmerhbnrca douglas graham date apr gmt sender newsbmerhbnrca usenet news
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
```

Processors : 8

```
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq atheist resources summary books addresses music a
xref altatheism newsanswers altanswers path mathew newsgroups subject altatheism faq introduction atheism summary please read file pos
newsgroups altatheism path idbsturztubsde benedikt rosenau subject gospel dating messageid sender mr nntp inews entry organization tec
xref altatheism path mathew newsgroups subject university violating separation churchstate messageid date mon apr gmt references aakep
xref altatheism socmotss recscouting newsgroups path rob strom subject socmotss et al princeton axes matching funds boy scouts sender
newsgroups altatheism path idbsturztubsde benedikt rosenau subject visit jehovahs witnesses messageid sender mr nntp inews entry organ
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references organization california
newsgroups altatheism path idbsturztubsde benedikt rosenau subject anecdote islam messageid sender mr nntp inews entry organization te
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu marbmerhbnrca km
path keith allan schneider newsgroups altatheism subject pompous ass messageid date apr gmt references mvspsuvmpsuedu organization cal
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
xref altatheism talkorigins path keith allan schneider newsgroups subject albert sabin messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject keith schneider stealth poster messageid date apr gmt references organization
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path keith allan schneider newsgroups altatheism subject political atheists messageid date apr gmt references viceicotekcom organizati
path newsgroups altatheism subject dont innocents die without death penalty messageid viceicotekcom robert beauchaine date apr gmt ref
path newsgroups altatheism subject ancient islamic rituals messageid viceicotekcom robert beauchaine date apr gmt references organizat
path newsgroups altatheism subject political atheists messageid viceicotekcom robert beauchaine date apr gmt references organization t
xref altatheism talkorigins newsgroups path scott mullins subject return abused creationist thread real probability abiogenesis messag
xref altatheism path newsgroups subject list biblical contradictions messageid bluecispittedu david joslin date apr gmt sender newspit
path newsgroups altatheism subject must creator maybe messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbea
path newsgroups altatheism subject americans evolution messageid ursabearcom halatpoohbears jim halat date apr gmt replyto halatpoohbe
path newsgroups altatheism subject speculations messageid aprbmerhbnrca douglas graham date apr gmt sender newsbmerhbnrca usenet news
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
path keith allan schneider newsgroups altatheism subject political atheists date apr gmt organization california institute technology
```

In [7]:

```
workers_time = {'P:2': 29.5806644000113,
                'P:4': 18.714486899989424,
                'P:6': 19.172850700007984,
                'P:8': 23.005617300019367
    }

time = pd.DataFrame.from_dict(workers_time, orient = 'index')
time.columns = ['time']
time = time.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time
```

Out[7]:

	time
P:2	29.580664
P:4	18.714487
P:6	19.172851
P:8	23.005617

From the table we can observe that for 2 workers the time taken was more than the rest and as the number of workers increased the time taken to complete the task was dropped. After a certain point if the number of workers are increased they lead to addition of more time needed to maintain communication among different workers.

Exercise 2: Calculate Term Frequency (TF)

The Term Frequency (TF) is calculated by counting the number of times a token occurs in the document. This TF score is relative to a specific document, therefore you need to normalized it by dividing with the total number of tokens appearing in the document. Develop an solution using MPI framework and write code. Please explain how you parallelize (or distribute) TF(t,d) calculation. Also explain your strategy from the data division and calculation division point of view as well. Perform a small experiment by varying number workers i.e. P = {2, 4, 8}. Also verify if you get the same result at the end.

The file from last part is first read and stored in a list of lists with each list containing words for individual text file. Then again the documents are divided among workers and each worker calculate the term frequency by counting the number of times a token appears in a document divided by the size of the document. Further the results from each worker are combined together and is stored in a text file which each row as a dictionary containing term frequency of tokens in that document.

In [ ]:

```
# Importing Libraries
import os
import numpy as np
from mpi4py import MPI

# Setting MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
set_files = None

startTime = MPI.Wtime()
# Master Node
if rank == 0:
    file = open("tokenized_data.txt", "r") # This file contains words of individual documen
    word_corpus = []
    for doc in file:
        word_doc = doc.split(' ')[:-1]
        word_corpus.append(word_doc)      # All the words are appended in a list for each d

    num_files = len(word_corpus)
    step = num_files//size

    set_files = []       # stores chunks containing words for each worker
    for i in range(size - 1):
        set_files.append(word_corpus[i*step: (i + 1)*step])
    set_files.append(word_corpus[(size - 1)*step: ])

worker_chunk = comm.scatter(set_files, root = 0)

term_frequency = []
for docs in worker_chunk:      # for each document in a worker chunk the term frequency of
    len_doc = len(docs)                              # are calculated
    token_freq = {}
    for word in docs:
        if word not in token_freq.keys():
            token_freq[word] = 1/len_doc
        else:
            token_freq[word] += 1/len_doc
    term_frequency.append(token_freq)

frequency_data = comm.gather(term_frequency, root = 0) # gathering term frequency data from

if rank == 0:
    term_doc = [item for sublist in frequency_data for item in sublist]
    with open('Termfrequency.txt', 'a') as f:
        for item in term_doc:
            f.write(str(item))
            f.write('\n')
    print('Time:', MPI.Wtime()-startTime)
```

Results of Term Frequency when used 2, 4 and 8 processors respectively.

Processors : 2

```
image.png{'xref': 0.0009407337723424271, 'altatheism': 0.0028222013170272815, 'newsanswers': 0.0009407337723424271, 'altanswers': 0.00
{'xref': 0.0003729951510630362, 'altatheism': 0.0022379709063782174, 'newsanswers': 0.0003729951510630362, 'altanswers': 0.0003729951
{'newsgroups': 0.0028169014084507044, 'altatheism': 0.0028169014084507044, 'path': 0.0028169014084507044, 'idbsturztubsde': 0.0028169
{'xref': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'mathew': 0.014492753623188406, 'news
{'xref': 0.012987012987012988, 'altatheism': 0.012987012987012988, 'socmotss': 0.025974025974025976, 'recscouting': 0.012987012987012
{'newsgroups': 0.002512562814070352, 'altatheism': 0.002512562814070352, 'path': 0.002512562814070352, 'idbsturztubsde': 0.00251256281
{'path': 0.02, 'keith': 0.04, 'allan': 0.02, 'schneider': 0.02, 'newsgroups': 0.02, 'altatheism': 0.02, 'subject': 0.02, 'political':
{'newsgroups': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'idbsturztubsde': 0.007246376811
{'path': 0.004651162790697674, 'keith': 0.013953488372093023, 'allan': 0.004651162790697674, 'schneider': 0.009302325581395349, 'newsg
{'path': 0.023255813953488372, 'keith': 0.06976744186046512, 'allan': 0.023255813953488372, 'schneider': 0.023255813953488372, 'newsgr
{'path': 0.01639344262295082, 'keith': 0.03278688524590164, 'allan': 0.01639344262295082, 'schneider': 0.01639344262295082, 'newsgroup
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgro
{'xref': 0.013888888888888888, 'altatheism': 0.013888888888888888, 'talkorigins': 0.013888888888888888, 'path': 0.013888888888888888,
{'path': 0.009345794392523364, 'keith': 0.028037383177570093, 'allan': 0.009345794392523364, 'schneider': 0.018691588785046728, 'newsgr
{'path': 0.0049261083743842365, 'keith': 0.009852216748768473, 'allan': 0.0049261083743842365, 'schneider': 0.0049261083743842365, 'ne
{'path': 0.0045045045045045045, 'keith': 0.009009009009009009, 'allan': 0.0045045045045045045, 'schneider': 0.0045045045045045045, 'ne
{'path': 0.007462686567164179, 'keith': 0.014925373134328358, 'allan': 0.007462686567164179, 'schneider': 0.007462686567164179, 'newsg
{'path': 0.01098901098901099, 'newsgroups': 0.01098901098901099, 'altatheism': 0.01098901098901099, 'subject': 0.01098901098901099, '
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009
{'path': 0.005780346820809248, 'newsgroups': 0.005780346820809248, 'altatheism': 0.005780346820809248, 'subject': 0.005780346820809248
{'xref': 0.003597122302158736, 'altatheism': 0.003597122302158736, 'talkorigins': 0.003597122302158736, 'newsgroups': 0.003597122302158
{'xref': 0.0049261083743842365, 'altatheism': 0.0049261083743842365, 'path': 0.0049261083743842365, 'newsgroups': 0.004926108374384236
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009
{'path': 0.022727272727272728, 'newsgroups': 0.022727272727272728, 'altatheism': 0.022727272727272728, 'subject': 0.022727272727272728
{'path': 0.011494252873563218, 'newsgroups': 0.011494252873563218, 'altatheism': 0.011494252873563218, 'subject': 0.011494252873563218
{'path': 0.0013679890560875513, 'keith': 0.0027359781121751026, 'allan': 0.0013679890560875513, 'schneider': 0.0013679890560875513, '
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgro
{'xref': 0.009009009009009009, 'altatheism': 0.009009009009009009, 'path': 0.009009009009009009, 'rmicacuk': 0.009009009009009009, 'm
{'newsgroups': 0.015151515151515152, 'altatheism': 0.015151515151515152, 'path': 0.015151515151515152, 'kilmanyfiuedu': 0.015151515151
```

Processors : 4

```
{'xref': 0.0009407337723424271, 'altatheism': 0.0028222013170272815, 'newsanswers': 0.0009407337723424271, 'altanswers': 0.000940733772
{'xref': 0.0003729951510630362, 'altatheism': 0.0022379709063782174, 'newsanswers': 0.0003729951510630362, 'altanswers': 0.000372995151
{'newsgroups': 0.0028169014084507044, 'altatheism': 0.0028169014084507044, 'path': 0.0028169014084507044, 'idbsturztubsde': 0.002816901
{'xref': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'mathew': 0.014492753623188406, 'news
{'xref': 0.012987012987012988, 'altatheism': 0.012987012987012988, 'socmotss': 0.025974025974025976, 'recscouting': 0.012987012987012
{'newsgroups': 0.002512562814070352, 'altatheism': 0.002512562814070352, 'path': 0.002512562814070352, 'idbsturztubsde': 0.002512562814
{'path': 0.02, 'keith': 0.04, 'allan': 0.02, 'schneider': 0.02, 'newsgroups': 0.02, 'altatheism': 0.02, 'subject': 0.02, 'political':
{'newsgroups': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'idbsturztubsde': 0.007246376811
{'path': 0.004651162790697674, 'keith': 0.013953488372093023, 'allan': 0.004651162790697674, 'schneider': 0.009302325581395349, 'newsg
{'path': 0.023255813953488372, 'keith': 0.06976744186046512, 'allan': 0.023255813953488372, 'schneider': 0.023255813953488372, 'newsgr
{'path': 0.01639344262295082, 'keith': 0.03278688524590164, 'allan': 0.01639344262295082, 'schneider': 0.01639344262295082, 'newsgroup
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgro
{'xref': 0.013888888888888888, 'altatheism': 0.013888888888888888, 'talkorigins': 0.013888888888888888, 'path': 0.013888888888888888,
{'path': 0.009345794392523364, 'keith': 0.028037383177570093, 'allan': 0.009345794392523364, 'schneider': 0.018691588785046728, 'newsgr
{'path': 0.0049261083743842365, 'keith': 0.009852216748768473, 'allan': 0.0049261083743842365, 'schneider': 0.0049261083743842365, 'ne
{'path': 0.0045045045045045045, 'keith': 0.009009009009009009, 'allan': 0.0045045045045045045, 'schneider': 0.0045045045045045045, 'ne
{'path': 0.007462686567164179, 'keith': 0.014925373134328358, 'allan': 0.007462686567164179, 'schneider': 0.007462686567164179, 'newsg
{'path': 0.01098901098901099, 'newsgroups': 0.01098901098901099, 'altatheism': 0.01098901098901099, 'subject': 0.01098901098901099, '
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009
{'path': 0.005780346820809248, 'newsgroups': 0.005780346820809248, 'altatheism': 0.005780346820809248, 'subject': 0.005780346820809248
{'xref': 0.003597122302158736, 'altatheism': 0.003597122302158736, 'talkorigins': 0.003597122302158736, 'newsgroups': 0.003597122302158
{'xref': 0.0049261083743842365, 'altatheism': 0.0049261083743842365, 'path': 0.0049261083743842365, 'newsgroups': 0.004926108374384236
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009
{'path': 0.022727272727272728, 'newsgroups': 0.022727272727272728, 'altatheism': 0.022727272727272728, 'subject': 0.022727272727272728
{'path': 0.011494252873563218, 'newsgroups': 0.011494252873563218, 'altatheism': 0.011494252873563218, 'subject': 0.011494252873563218
{'path': 0.0013679890560875513, 'keith': 0.0027359781121751026, 'allan': 0.0013679890560875513, 'schneider': 0.0013679890560875513, '
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgro
{'xref': 0.009009009009009009, 'altatheism': 0.009009009009009009, 'path': 0.009009009009009009, 'rmicacuk': 0.009009009009009009, 'm
{'newsgroups': 0.015151515151515152, 'altatheism': 0.015151515151515152, 'path': 0.015151515151515152, 'kilmanyfiuedu': 0.015151515151
```

Processors : 8

```
{'xref': 0.0009407337723424271, 'altatheism': 0.0028222013170272815, 'newsanswers': 0.0009407337723424271, 'altanswers': 0.000940733772
{'xref': 0.0003729951510630362, 'altatheism': 0.0022379709063782174, 'newsanswers': 0.0003729951510630362, 'altanswers': 0.000372995151
{'newsgroups': 0.0028169014084507044, 'altatheism': 0.0028169014084507044, 'path': 0.0028169014084507044, 'idbsturztubsde': 0.002816901
{'xref': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'mathew': 0.014492753623188406, 'newsg
{'xref': 0.012987012987012988, 'altatheism': 0.012987012987012988, 'socmotss': 0.025974025974025976, 'recscouting': 0.012987012987012988
{'newsgroups': 0.002512562814070352, 'altatheism': 0.002512562814070352, 'path': 0.002512562814070352, 'idbsturztubsde': 0.002512562814
{'path': 0.02, 'keith': 0.04, 'allan': 0.02, 'schneider': 0.02, 'newsgroups': 0.02, 'altatheism': 0.02, 'subject': 0.02, 'political': 0
{'newsgroups': 0.007246376811594203, 'altatheism': 0.007246376811594203, 'path': 0.007246376811594203, 'idbsturztubsde': 0.007246376811
{'path': 0.004651162790697674, 'keith': 0.013953488372093023, 'allan': 0.004651162790697674, 'schneider': 0.009302325581395349, 'newsgr
{'path': 0.023255813953488372, 'keith': 0.06976744186046512, 'allan': 0.023255813953488372, 'schneider': 0.023255813953488372, 'newsgro
{'path': 0.01639344262295082, 'keith': 0.03278688524590164, 'allan': 0.01639344262295082, 'schneider': 0.01639344262295082, 'newsgroups
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgrou
{'xref': 0.013888888888888888, 'altatheism': 0.013888888888888888, 'talkorigins': 0.013888888888888888, 'path': 0.013888888888888888, '
{'path': 0.009345794392523364, 'keith': 0.028037383177570093, 'allan': 0.009345794392523364, 'schneider': 0.018691588785046728, 'newsgro
{'path': 0.0049261083743842365, 'keith': 0.009852216748768473, 'allan': 0.0049261083743842365, 'schneider': 0.0049261083743842365, 'new
{'path': 0.0045045045045045045, 'keith': 0.009009009009009009, 'allan': 0.0045045045045045045, 'schneider': 0.0045045045045045045, 'new
{'path': 0.007462686567164179, 'keith': 0.014925373134328358, 'allan': 0.007462686567164179, 'schneider': 0.007462686567164179, 'newsgr
{'path': 0.01098901098901099, 'newsgroups': 0.01098901098901099, 'altatheism': 0.01098901098901099, 'subject': 0.01098901098901099, 'do
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009,
{'path': 0.005780346820809248, 'newsgroups': 0.005780346820809248, 'altatheism': 0.005780346820809248, 'subject': 0.005780346820809248,
{'xref': 0.003597122302158273, 'altatheism': 0.003597122302158273, 'talkorigins': 0.003597122302158273, 'newsgroups': 0.003597122302
{'xref': 0.0049261083743842365, 'altatheism': 0.0049261083743842365, 'path': 0.0049261083743842365, 'newsgroups': 0.0049261083743842365
{'path': 0.008130081300813009, 'newsgroups': 0.008130081300813009, 'altatheism': 0.008130081300813009, 'subject': 0.008130081300813009,
{'path': 0.022727272727272728, 'newsgroups': 0.022727272727272728, 'altatheism': 0.022727272727272728, 'subject': 0.022727272727272728,
{'path': 0.011494252873563218, 'newsgroups': 0.011494252873563218, 'altatheism': 0.011494252873563218, 'subject': 0.011494252873563218,
{'path': 0.0013679890560875513, 'keith': 0.0027359781121751026, 'allan': 0.0013679890560875513, 'schneider': 0.0013679890560875513, 'ne
{'path': 0.017543859649122806, 'keith': 0.05263157894736842, 'allan': 0.017543859649122806, 'schneider': 0.03508771929824561, 'newsgrou
{'xref': 0.009009009009009009, 'altatheism': 0.009009009009009009, 'path': 0.009009009009009009, 'rmicacuk': 0.009009009009009009, 'mr'
{'newsgroups': 0.015151515151515152, 'altatheism': 0.015151515151515152, 'path': 0.015151515151515152, 'kilmanyfiuedu': 0.015151515151
```


In [8]:

```
workers_time2 = {'P:2': 7.300251100008609,
                 'P:4': 6.451950099988608,
                 'P:6': 6.068260400003055,
                 'P:8': 6.02938780002296
    }

time2 = pd.DataFrame.from_dict(workers_time2, orient = 'index')
time2.columns = ['time']
time2 = time2.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time2
```

Out[8]:

	time
P:2	7.300251
P:4	6.451950
P:6	6.068260
P:8	6.029388

Exercise 3: Calculate Inverse Document Frequency

The Inverse Document Frequency ((IDF) is counting the number of documents in the corpus and counting the number of documents that contain a token. While the TF is calculated on a per-document basis, the IDF is computed on the basis of the entire corpus. The final IDF score of a token t in the corpus C is obtained by taking logarithm of document count in the corpus divided by the number of documents in the corpus that contain a particular token.

Develop an solution using MPI framework and write cod. Please explain how you parallelize (or distribute) IDF(t,d) calculation. Also explain your strategy from the data division and calculation division point of view as well. Perform a small experiment by varying number workers i.e. P = {2, 4, 8}. Also verify if you get the same result at the end.

The IDF is calculated for the tokens in the entire corpus by dividing the size of the corpus by the number of documents where that token appears and then taking the log. The smaller the IDF, the more frequent that token appears and vice versa.

Each worker calculates the number of documents where that token appears for the corpus chunk it has recieved and then when every chunk from each worker is gathered the values corresponding to each token in the dictionaries are added together which gives the total number of documents where the token appears in the entire corpus.

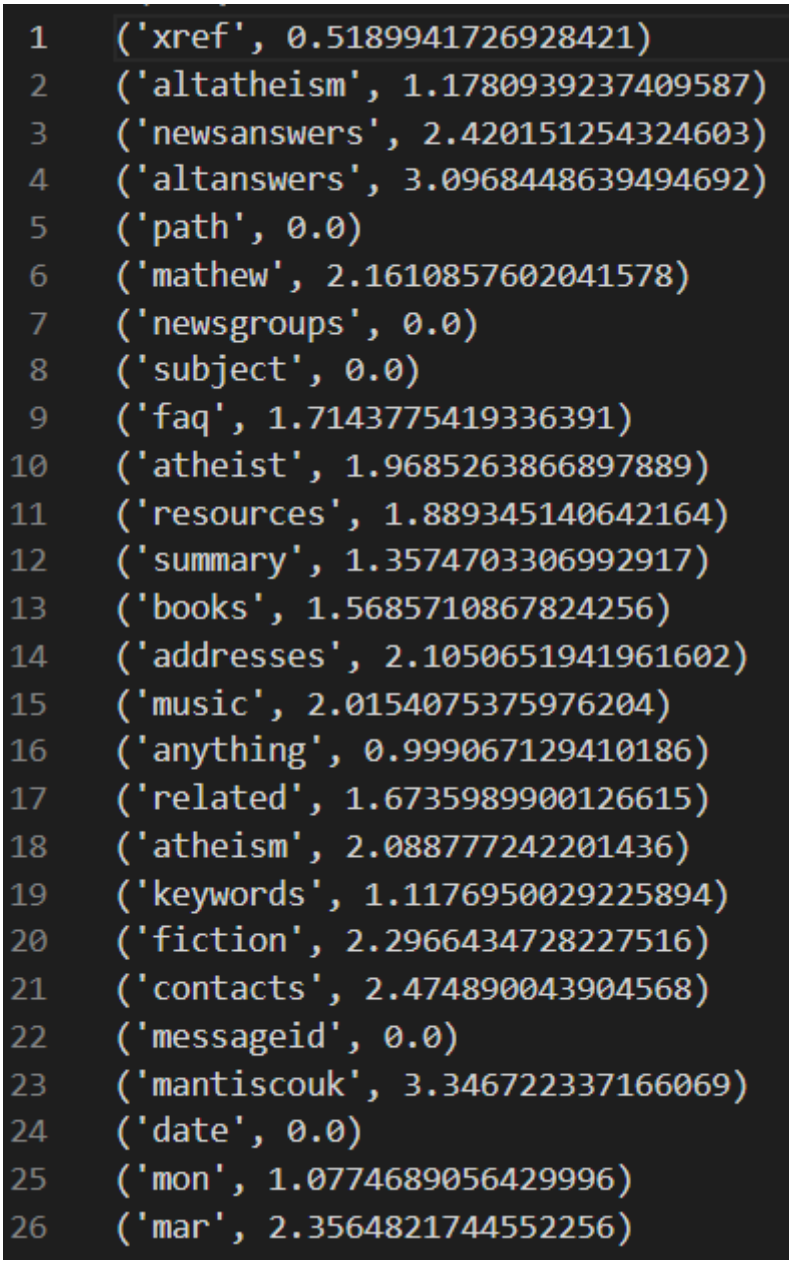
In [ ]:

```
# Importing Libraries
import math
import os
import numpy as np
from mpi4py import MPI
import ast

# Setting MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

set_files = None

startTime = MPI.Wtime()
# Master node
if rank == 0:
    file = open("Termfrequency.txt", "r") # File containing dictionaries which contains ter
    doc_termfreq = []
    for doc in file:
        doc_termfreq.append(ast.literal_eval(doc)) # append all dictionaries as items in a

    corpus_size = len(doc_termfreq)    # Corpus size(number of documents)
    step = corpus_size//size

    set_files = []                 # stores chunks containing dictionaries for each worker
    for i in range(size - 1):
        set_files.append(doc_termfreq[i*step: (i + 1)*step])
    set_files.append(doc_termfreq[(size - 1)*step: ])

worker_chunk = comm.scatter(set_files, root = 0)

token_count = {}           # Each worker calculates in how many documents it has recieved i
for dict in worker_chunk:
    for token in dict.keys():
        if token not in token_count.keys():
            token_count[token] = 1
        else:
            token_count[token] += 1

count_data = comm.gather(token_count, root = 0) # gathering dictionaries from each worker c
                                                # number of documents where token a

if rank == 0:   # The master calculates for the entire corpus
    inv_termfreq = {}
    for dict in count_data:   # contains list of dictionaries
        for key in dict.keys():
            if key not in inv_termfreq.keys():
                inv_termfreq[key] = dict[key]
            else:
                inv_termfreq[key] += dict[key]

    IDF = {}                                  # Calculates IDF
    for k, v in inv_termfreq.items():
        IDF[k] = math.log10(corpus_size/v)
    with open('InvTermfrequency.txt', 'a') as f:         #Saving results to a text file
        for item in IDF.items():                         # Each line of file contains tupple
            f.write(str(item))
            f.write('\n')
```

```
print('Time:', MPI.Wtime()-startTime)
```

Results of IDF when used 2, 4 and 8 processors respectively.

Processors : 2



```
('xref', 0.5189941726928421)
('altatheism', 1.1780939237409587)
('newsanswers', 2.420151254324603)
('altanswers', 3.0968448639494692)
('path', 0.0)
('mathew', 2.1610857602041578)
('newsgroups', 0.0)
('subject', 0.0)
('faq', 1.7143775419336391)
('atheist', 1.9685263866897889)
('resources', 1.889345140642164)
('summary', 1.3574703306992917)
('books', 1.5685710867824256)
('addresses', 2.1050651941961602)
('music', 2.0154075375976204)
('anything', 0.999067129410186)
('related', 1.6735989900126615)
('atheism', 2.088777242201436)
('keywords', 1.1176950029225894)
('fiction', 2.2966434728227516)
('contacts', 2.474890043904568)
('messageid', 0.0)
('mantiscouk', 3.346722337166069)
('date', 0.0)
('mon', 1.0774689056429996)
('mar', 2.3564821744552256)
```

Processors : 4

```
('xref', 0.5189941726928421)
('altatheism', 1.1780939237409587)
('newsanswers', 2.420151254324603)
('altanswers', 3.0968448639494692)
('path', 0.0)
('mathew', 2.1610857602041578)
('newsgroups', 0.0)
('subject', 0.0)
('faq', 1.7143775419336391)
('atheist', 1.9685263866897889)
('resources', 1.889345140642164)
('summary', 1.3574703306992917)
('books', 1.5685710867824256)
('addresses', 2.1050651941961602)
('music', 2.0154075375976204)
('anything', 0.999067129410186)
('related', 1.6735989900126615)
('atheism', 2.088777242201436)
('keywords', 1.1176950029225894)
('fiction', 2.2966434728227516)
('contacts', 2.474890043904568)
('messageid', 0.0)
('mantiscouk', 3.346722337166069)
('date', 0.0)
('mon', 1.0774689056429996)
('mar', 2.3564821744552256)
```

Processors : 8

```
('xref', 0.5189941726928421)
('altatheism', 1.1780939237409587)
('newsanswers', 2.420151254324603)
('altanswers', 3.0968448639494692)
('path', 0.0)
('mathew', 2.1610857602041578)
('newsgroups', 0.0)
('subject', 0.0)
('faq', 1.7143775419336391)
('atheist', 1.9685263866897889)
('resources', 1.889345140642164)
('summary', 1.3574703306992917)
('books', 1.5685710867824256)
('addresses', 2.1050651941961602)
('music', 2.0154075375976204)
('anything', 0.999067129410186)
('related', 1.6735989900126615)
('atheism', 2.088777242201436)
('keywords', 1.1176950029225894)
('fiction', 2.2966434728227516)
('contacts', 2.474890043904568)
('messageid', 0.0)
('mantiscouk', 3.346722337166069)
('date', 0.0)
('mon', 1.0774689056429996)
('mar', 2.3564821744552256)
```

In [9]:

```
workers_time3 = {'P:2': 14.346128000004683,
                 'P:4': 13.785756299999775,
                 'P:6': 14.655860300001223,
                 'P:8': 14.441067200008547
    }

time3 = pd.DataFrame.from_dict(workers_time3, orient = 'index')
time3.columns = ['time']
time3 = time3.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time3
```

Out[9]:

	time
P:2	14.346128
P:4	13.785756
P:6	14.655860
P:8	14.441067

Calculate Term Frequency Inverse Document Frequency (TF-IDF) scores

In this exercise you will fnd the TF-IDF(t,d) for a given token t and a document d in the corpus C is the product of TF(t,d) and IDF(t), which is represented as, TF-IDF(t, d) = TF(t, d) * IDF(t) ,

You have already calculated TF(t, d) in Exercise 2 and IDF(t) is Exercise 3.

In this exercise you have to think about how you can combine the complete pipeline in a single parallel/distributed program that can run worker P = {2, 4, 8}. Develop an solution using MPI framework and write code. Please explain how you parallelize (or distribute) the complete pipeline. Also explain your strategy from the data division and calculation division point of view as well. Perform a small experiment by varying number workers i.e. P = {2, 4, 8}. Also verify if you get the same result at the end.

Finally we have two files one containing the Term Frequency and other containing Inverse Term frequency. The file containing term frequencies is splitted and scattered among all workers and the other one is broadcasted to all workers. Then for each token in each document the Tfldf is calculated by

mutiplying the Tf and Idf the workers have and in the end the results are gathered and again stored in a file for displaying.

In [ ]:

```
# Importing Libraries
import os
import numpy as np
from mpi4py import MPI
import ast

# Setting MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()
set_files = None
Idf = None
idf_corpus = None

startTime = MPI.Wtime()
# Master Node
if rank == 0:
    file1 = open("Termfrequency.txt", "r")     # stores Term Frequency
    file2 = open("InvTermfrequency.txt", "r")                 # stores Inverse term Frequen
    doc_termfreq = []  # List containing dictionaries having term frequencies
    for doc in file1:
        doc_termfreq.append(ast.literal_eval(doc))

    step = len(doc_termfreq)//size
    set_files = []     # stores chunks containing dictionaries for each worker
    for i in range(size - 1):
        set_files.append(doc_termfreq[i*step: (i + 1)*step])
    set_files.append(doc_termfreq[(size - 1)*step: ])

    idf_corpus = []
    for tup in file2:
        idf_corpus.append(ast.literal_eval(tup))
    idf_corpus = dict(idf_corpus)             # dictionary containing IDF for entire corpus

Idf = comm.bcast(idf_corpus, root = 0)            # IDF is broadcasted to each worker
worker_chunk = comm.scatter(set_files, root = 0)      # Dictionaries containing term freque

Tf_Idf = {}
for i, dict in enumerate(worker_chunk):
    token_count = {}                          # TF and IDF are multiplied together to get TF
    for token in dict.keys():
        if token in Idf.keys():
            token_count[token] = dict[token]*Idf[token]
    Tf_Idf['doc' + str(rank*(len(worker_chunk) - 1) + i)] = token_count

tfidf_data = comm.gather(Tf_Idf, root = 0)

if rank == 0:
    TFIDF = {}           # adding all dictionaries to one big dictionary from all workers
    for d in tfidf_data:
        TFIDF.update(d)
    with open('TfIdf.txt', 'a') as f:     # Saving to a text file where each row contains i
        for item in TFIDF.items():
            f.write(str(item))
            f.write('\n')
    print('Time:', MPI.Wtime()-startTime)
```

Results of TF-IDF when used 2, 4 and 8 processors respectively.

Processors : 2

```
('doc0', {'xref': 0.00048823534590107444, 'altatheism': 0.003324818223163571, 'newsanswers': 0.0022767180191200402, 'altanswers': 0.00
('doc1', {'xref': 0.00019358230984440215, 'altatheism': 0.002636539926313224, 'newsanswers': 0.0009027046827022018, 'altanswers': 0.00
('doc2', {'newsgroups': 0.0, 'altatheism': 0.0033185744330731233, 'path': 0.0, 'idbsturztubsde': 0.007169267580092684, 'benedikt': 0.0
('doc3', {'xref': 0.003760827338353928, 'altatheism': 0.008536912490876512, 'path': 0.0, 'mathew': 0.03132008348121968, 'newsgroups': 
('doc4', {'xref': 0.0067401840609460015, 'altatheism': 0.015299921087544919, 'socmotss': 0.07038701920984143, 'recscouting': 0.041841
('doc5', {'newsgroups': 0.0, 'altatheism': 0.0029600349842737656, 'path': 0.0, 'idbsturztubsde': 0.0063946984696806606, 'benedikt': 0.
('doc6', {'path': 0.0, 'keith': 0.0678263920185714, 'allan': 0.043096736218543125, 'schneider': 0.04450835770425726, 'newsgroups': 0.0
('doc7', {'newsgroups': 0.0, 'altatheism': 0.008536912490876512, 'path': 0.0, 'idbsturztubsde': 0.018442681093716686, 'benedikt': 0.03
('doc8', {'path': 0.0, 'keith': 0.023660369308803973, 'allan': 0.010022496795010029, 'schneider': 0.020701561722910357, 'newsgroups': 
('doc9', {'path': 0.0, 'keith': 0.11830184654401986, 'allan': 0.05011248397505014, 'schneider': 0.05175390430727589, 'newsgroups': 0.0
('doc10', {'path': 0.0, 'keith': 0.055595403293910974, 'allan': 0.03532519362175666, 'schneider': 0.036482260413325626, 'newsgroups': 
('doc11', {'path': 0.0, 'keith': 0.08924525265601498, 'allan': 0.037804154577669405, 'schneider': 0.07808483807764433, 'newsgroups': 0
('doc12', {'xref': 0.007208252398511695, 'altatheism': 0.016362415607513316, 'talkorigins': 0.02640682118228272, 'path': 0.0, 'keith': 
('doc13', {'path': 0.0, 'keith': 0.04754186356441919, 'allan': 0.020138661784365942, 'schneider': 0.04159659598528716, 'newsgroups': 0
('doc14', {'path': 0.0, 'keith': 0.016706007886347633, 'allan': 0.010614959659739685, 'schneider': 0.010962649680851543, 'newsgroups': 
('doc15', {'path': 0.0, 'keith': 0.01527621441859715 8, 'allan': 0.009706472121293495, 'schneider': 0.01002440488834623, 'newsgroups': 
('doc16', {'path': 0.0, 'keith': 0.02530835523081022, 'allan': 0.016080871723336984, 'schneider': 0.016607596158304948, 'newsgroups': 
('doc17', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.01294608707407647, 'subject': 0.0, 'dont': 0.005814104685882367, 'innocents
('doc18', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'ancient': 0.01615526525395318, 'islami
('doc19', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.006809791466710744, 'subject': 0.0, 'political': 0.00898007536831775, 'athe
('doc20', {'xref': 0.0018668855132836048, 'altatheism': 0.004237747927125751, 'talkorigins': 0.006839176709080417, 'newsgroups': 0.0, 
('doc21', {'xref': 0.0025566215403588282, 'altatheism': 0.005803418343551521, 'path': 0.0, 'newsgroups': 0.0, 'subject': 0.0, 'list': 
('doc22', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'must': 0.008136620403691397, 'creator'
('doc23', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.026774861903203606, 'subject': 0.0, 'americans': 0.04155019084745868, 'evol
('doc24', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.013541309468286881, 'subject': 0.0, 'speculations': 0.03473806029485707, 'm
```

Processors : 4

```
('doc0', {'xref': 0.00048823534590107444, 'altatheism': 0.003324818223163571, 'newsanswers': 0.0022767180191200402, 'altanswers': 0.002
('doc1', {'xref': 0.00019358230984440215, 'altatheism': 0.002636539926313224, 'newsanswers': 0.0009027046827022018, 'altanswers': 0.001
('doc2', {'newsgroups': 0.0, 'altatheism': 0.0033185744330731233, 'path': 0.0, 'idbsturztubsde': 0.007169267580092684, 'benedikt': 0.01
('doc3', {'xref': 0.003760827338353928, 'altatheism': 0.008536912490876512, 'path': 0.0, 'mathew': 0.03132008348121968, 'newsgroups': 0
('doc4', {'xref': 0.0067401840609460015, 'altatheism': 0.015299921087544919, 'socmotss': 0.07038701920984143, 'recscouting': 0.04184134
('doc5', {'newsgroups': 0.0, 'altatheism': 0.0029600349842737656, 'path': 0.0, 'idbsturztubsde': 0.0063946984696806606, 'benedikt': 0.0
('doc6', {'path': 0.0, 'keith': 0.0678263920185714, 'allan': 0.043096736218543125, 'schneider': 0.04450835770425726, 'newsgroups': 0.0,
('doc7', {'newsgroups': 0.0, 'altatheism': 0.008536912490876512, 'path': 0.0, 'idbsturztubsde': 0.018442681093716686, 'benedikt': 0.035
('doc8', {'path': 0.0, 'keith': 0.023660369308803973, 'allan': 0.010022496795010029, 'schneider': 0.020701561722910357, 'newsgroups': 0
('doc9', {'path': 0.0, 'keith': 0.11830184654401986, 'allan': 0.05011248397505014, 'schneider': 0.05175390430727589, 'newsgroups': 0.0,
('doc10', {'path': 0.0, 'keith': 0.055595403293910974, 'allan': 0.03532519362175666, 'schneider': 0.036482260413325626, 'newsgroups': 0
('doc11', {'path': 0.0, 'keith': 0.08924525265601498, 'allan': 0.037804154577669405, 'schneider': 0.07808483807764433, 'newsgroups': 0.
('doc12', {'xref': 0.007208252398511695, 'altatheism': 0.016362415607513316, 'talkorigins': 0.02640682118228272, 'path': 0.0, 'keith': 
('doc13', {'path': 0.0, 'keith': 0.04754186356441919, 'allan': 0.020138661784365942, 'schneider': 0.04159659598528716, 'newsgroups': 0.
('doc14', {'path': 0.0, 'keith': 0.016706007886347633, 'allan': 0.010614959659739685, 'schneider': 0.010962649680851543, 'newsgroups': 
('doc15', {'path': 0.0, 'keith': 0.015276214418597158, 'allan': 0.009706472121293495, 'schneider': 0.01002440488834623, 'newsgroups': 0
('doc16', {'path': 0.0, 'keith': 0.02530835523081022, 'allan': 0.016080871723336984, 'schneider': 0.016607596158304948, 'newsgroups': 0
('doc17', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.01294608707407647, 'subject': 0.0, 'dont': 0.005814104685882367, 'innocents'
('doc18', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'ancient': 0.01615526525395318, 'islamic
('doc19', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.006809791466710744, 'subject': 0.0, 'political': 0.00898007536831775, 'athei
('doc20', {'xref': 0.0018668855132836048, 'altatheism': 0.004237747927125751, 'talkorigins': 0.006839176709080417, 'newsgroups': 0.0, '
('doc21', {'xref': 0.0025566215403588282, 'altatheism': 0.005803418343551521, 'path': 0.0, 'newsgroups': 0.0, 'subject': 0.0, 'list': 0
('doc22', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'must': 0.008136620403691397, 'creator':
('doc23', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.026774861903203606, 'subject': 0.0, 'americans': 0.04155019084745868, 'evolu
('doc24', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.013541309468286881, 'subject': 0.0, 'speculations': 0.03473806029485707, 'me
('doc25', {'path': 0.0, 'keith': 0.004639288099765485, 'allan': 0.0029477931750029494, 'schneider': 0.0030443473121926994, 'newsgroups'
```

Processors : 8

```
('doc0', {'xref': 0.00048823534590107444, 'altatheism': 0.003324818223163571, 'newsanswers': 0.0022767180191200402, 'altanswers': 0.00
('doc1', {'xref': 0.00019358230984440215, 'altatheism': 0.002636539926313224, 'newsanswers': 0.0009027046827022018, 'altanswers': 0.00
('doc2', {'newsgroups': 0.0, 'altatheism': 0.003318574433073123, 'path': 0.0, 'idbsturztubsde': 0.007169267580092684, 'benedikt': 0.0
('doc3', {'xref': 0.003760827338353928, 'altatheism': 0.008536912490876512, 'path': 0.0, 'mathew': 0.03132008348121968, 'newsgroups':
('doc4', {'xref': 0.0067401840609460015, 'altatheism': 0.015299921087544919, 'socmotss': 0.07038701920984143, 'recscouting': 0.041841
('doc5', {'newsgroups': 0.0, 'altatheism': 0.0029600349842737656, 'path': 0.0, 'idbsturztubsde': 0.0063946984696806606, 'benedikt': 0.
('doc6', {'path': 0.0, 'keith': 0.0678263920185714, 'allan': 0.043096736218543125, 'schneider': 0.04450835770425726, 'newsgroups': 0.0
('doc7', {'newsgroups': 0.0, 'altatheism': 0.008536912490876512, 'path': 0.0, 'idbsturztubsde': 0.018442681093716686, 'benedikt': 0.03
('doc8', {'path': 0.0, 'keith': 0.023660369308803973, 'allan': 0.010022496795010029, 'schneider': 0.020701561722910357, 'newsgroups':
('doc9', {'path': 0.0, 'keith': 0.11830184654401986, 'allan': 0.05011248397505014, 'schneider': 0.05175390430727589, 'newsgroups': 0.0
('doc10', {'path': 0.0, 'keith': 0.055595403293910974, 'allan': 0.03532519362175666, 'schneider': 0.036482260413325626, 'newsgroups':
('doc11', {'path': 0.0, 'keith': 0.08924525265601498, 'allan': 0.037804154577669405, 'schneider': 0.07808483807764433, 'newsgroups': 0
('doc12', {'xref': 0.007208252398511695, 'altatheism': 0.016362415607513316, 'talkorigins': 0.02640682118228272, 'path': 0.0, 'keith':
('doc13', {'path': 0.0, 'keith': 0.04754186356441919, 'allan': 0.020138661784365942, 'schneider': 0.04159659598528716, 'newsgroups': 0
('doc14', {'path': 0.0, 'keith': 0.016706007886347633, 'allan': 0.010614959659739685, 'schneider': 0.010962649680851543, 'newsgroups':
('doc15', {'path': 0.0, 'keith': 0.015276214418597158, 'allan': 0.009706472121293495, 'schneider': 0.01002440488834623, 'newsgroups':
('doc16', {'path': 0.0, 'keith': 0.02530835523081022, 'allan': 0.016080871723336984, 'schneider': 0.016607596158304948, 'newsgroups':
('doc17', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.01294608707407647, 'subject': 0.0, 'dont': 0.005814104685882367, 'innocents
('doc18', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'ancient': 0.01615526525395318, 'islami
('doc19', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.006809791466710744, 'subject': 0.0, 'political': 0.00898007536831775, 'athe
('doc20', {'xref': 0.0018668855132836048, 'altatheism': 0.004237747927125751, 'talkorigins': 0.006839176709080417, 'newsgroups': 0.0,
('doc21', {'xref': 0.0025566215403588282, 'altatheism': 0.005803418343551521, 'path': 0.0, 'newsgroups': 0.0, 'subject': 0.0, 'list':
('doc22', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.009577999380007795, 'subject': 0.0, 'must': 0.008136620403691397, 'creator'
('doc23', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.026774861903203606, 'subject': 0.0, 'americans': 0.04155019084745868, 'evol
('doc24', {'path': 0.0, 'newsgroups': 0.0, 'altatheism': 0.013541309468286881, 'subject': 0.0, 'speculations': 0.03473806029485707, 'm
('doc25', {'path': 0.0, 'keith': 0.004639288099765485, 'allan': 0.0029477931750029494, 'schneider': 0.0030443473121926994, 'newsgroups
```

In [10]:

```
workers_time4 = {'P:2': 18.85017330001574,
                 'P:4': 18.260608500015223,
                 'P:6': 19.26716909999959,
                 'P:8': 18.903702600015095
    }

time4 = pd.DataFrame.from_dict(workers_time4, orient = 'index')
time4.columns = ['time']
time4 = time4.style.set_properties(**{
    'background-color': 'grey',
    'font-size': '20pt',
})
time4
```

Out[10]:

	time
P:2	18.850173
P:4	18.260609
P:6	19.267169
P:8	18.903703