CSE 508: IR ASSIGNMENT 1

(JAY SARAF 2020438)

1. Data Preprocessing: I have followed the 5 preprocessing steps in order as mentioned in the question. For lower casing, I have used python's inbuilt function. For tokenization, I have used NLTK library's word_tokenize, which is present in nltk.tokenize. For stopwords removal, I have used nltk.download for English language.

```
File 1 before preprocessing:
Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge
and want the most out of your springs than these are the way to go.

[Step a] : After Lower casing the text
loving these vintage springs on my vintage strat. they have a good tension and great stability. if you are floating your bridge
and want the most out of your springs than these are the way to go.
[Step b] : After tokenizing the text
['loving', 'these', 'vintage', 'springs', 'on', 'my', 'vintage', 'strat', '.', 'they', 'have', 'a', 'good', 'tension', 'and',
'great', 'stability', '.', 'if', 'you', 'are', 'floating', 'your', 'bridge', 'and', 'want', 'the', 'most', 'out', 'of', 'your',
'springs', 'than', 'these', 'are', 'the', 'way', 'to', 'go', '.']
[Step c] : After removing the stopwords from the text
['loving', 'vintage', 'springs', 'vintage', 'strat', '.', 'good', 'tension', 'great', 'stability', '.', 'floating', 'bridge',
'want', 'springs', 'way', 'go', '.']
[Step d] : After removing punctuations from text
['loving', 'vintage', 'springs', 'vintage', 'strat', 'good', 'tension', 'great', 'stability', 'floating', 'bridge', 'want', 'sp
rings', 'way', 'go', '']
[Step e] : After removing blank spaces from text
['loving', 'vintage', 'springs', 'vintage', 'strat', 'good', 'tension', 'great', 'stability', 'floating', 'bridge', 'want', 'sp
rings', 'way', 'go']
File 1 after preprocessing:
loving vintage springs vintage strat good tension great stability floating bridge want springs way go

   File 2 before preprocessing:
   Works great as a guitar bench mat. Not rugged enough for abuse but if you take care of it, it will take care of you. Makes orga
   nization of workspace much easier because screws won't roll around. Color is good too.

   [Step a] : After Lower casing the text
   works great as a guitar bench mat. not rugged enough for abuse but if you take care of it, it will take care of you. makes orga
   nization of workspace much easier because screws won't roll around. color is good too.
   [Step b] : After tokenizing the text
   ['works', 'great', 'as', 'a', 'guitar', 'bench', 'mat', '.', 'not', 'rugged', 'enough', 'for', 'abuse', 'but', 'if', 'you', 'ta
   ke', 'care', 'of', 'it', ',', 'it', 'will', 'take', 'care', 'of', 'you', '.', 'makes', 'organization', 'of', 'workspace', 'muc
   h', 'easier', 'because', 'screws', 'wo', "n't", 'roll', 'around', '.', 'color', 'is', 'good', 'too', '.']
   [Step c] : After removing the stopwords from the text
   ['works', 'great', 'guitar', 'bench', 'mat', '.', 'rugged', 'enough', 'abuse', 'take', 'care', ',', 'take', 'care', '.', 'make
   s', 'organization', 'workspace', 'much', 'easier', 'screws', 'wo', "n't", 'roll', 'around', '.', 'color', 'good', '.']
   [Step d] : After removing punctuations from text
   ['works', 'great', 'guitar', 'bench', 'mat', 'rugged', 'enough', 'abuse', 'take', 'care', 'take', 'care', 'makes', 'organizatio
   n', 'workspace', 'much', 'easier', 'screws', 'wo', 'n', 't', 'roll', 'around', 'color', 'good', '']
   [Step e] : After removing blank spaces from text
   ['works', 'great', 'guitar', 'bench', 'mat', 'rugged', 'enough', 'abuse', 'take', 'care', 'take', 'care', 'makes', 'organizatio
   n', 'workspace', 'much', 'easier', 'screws', 'wo', 'n', 't', 'roll', 'around', 'color', 'good']
   File 2 after preprocessing:
   works great guitar bench mat rugged enough abuse take care take care makes organization workspace much easier screws wo n t rol
   l around color good
```

File 3 before preprocessing:
We use these for everything from our acoustic bass down to our ukuleles. I know there is a smaller model available for ukes, vi
olins, etc.; we haven't yet ordered those, but these will work on smaller instruments if one doesn't extend the feet to their m
aximum width. They're gentle on the instruments, and the grippy material keeps them secure.

The greatest benefit has been when writing music at the computer and needing to set a guitar down to use the keyboard/mouse - j
ust easier for me than a hanging stand.

We have several and gave one to a friend for Christmas as well. I've used mine on stage, and it folds up small enough to fit ri
ght in my gig bag.

[Step a] : After Lower casing the text
we use these for everything from our acoustic bass down to our ukuleles. i know there is a smaller model available for ukes, vi
olins, etc.; we haven't yet ordered those, but these will work on smaller instruments if one doesn't extend the feet to their m
aximum width. they're gentle on the instruments, and the grippy material keeps them secure.

the greatest benefit has been when writing music at the computer and needing to set a guitar down to use the keyboard/mouse - j
ust easier for me than a hanging stand.

we have several and gave one to a friend for christmas as well. i've used mine on stage, and it folds up small enough to fit ri
ght in my gig bag.
[Step b] : After tokenizing the text
['we', 'use', 'these', 'for', 'everything', 'from', 'our', 'acoustic', 'bass', 'down', 'to', 'our', 'ukuleles', '.', 'i', 'kno
w', 'there', 'is', 'a', 'smaller', 'model', 'available', 'for', 'ukes', ',', 'violins', ',', 'etc', '.', ';', 'we', 'have',
"n't", 'yet', 'ordered', 'those', ',', 'but', 'these', 'will', 'work', 'on', 'smaller', 'instruments', 'if', 'one', 'does',
"n't", 'extend', 'the', 'feet', 'to', 'their', 'maximum', 'width', '.', 'they', "'re", 'gentle', 'on', 'the', 'instruments',
',', 'and', 'the', 'grippy', 'material', 'keeps', 'them', 'secure', '.', 'the', 'greatest', 'benefit', 'has', 'been', 'when',
'writing', 'music', 'at', 'the', 'computer', 'and', 'needing', 'to', 'set', 'a', 'guitar', 'down', 'to', 'use', 'the', 'keyboar
d/mouse', '-', 'just', 'easier', 'for', 'me', 'than', 'a', 'hanging', 'stand', '.', 'we', 'have', 'several', 'and', 'gave', 'on
e', 'to', 'a', 'friend', 'for', 'christmas', 'as', 'well', '.', 'i', "'ve", 'used', 'mine', 'on', 'stage', ',', 'and', 'it', 'f
olds', 'up', 'small', 'enough', 'to', 'fit', 'right', 'in', 'my', 'gig', 'bag', '.']
[Step c] : After removing the stopwords from the text
['use', 'everything', 'acoustic', 'bass', 'ukuleles', '.', 'know', 'smaller', 'model', 'available', 'ukes', ',', 'violins',
',', 'etc', '.', ';', "n't", 'yet', 'ordered', ',', 'work', 'smaller', 'instruments', 'one', "n't", 'extend', 'feet', 'maximu
m', 'width', '.', "'re", 'gentle', 'instruments', ',', 'grippy', 'material', 'keeps', 'secure', '.', 'greatest', 'benefit', 'wr
iting', 'music', 'computer', 'needing', 'set', 'guitar', 'use', 'keyboard/mouse', '-', 'easier', 'hanging', 'stand', '.', 'seve
ral', 'gave', 'one', 'friend', 'christmas', 'well', '.', "'ve", 'used', 'mine', 'stage', ',', 'folds', 'small', 'enough', 'fi
t', 'right', 'gig', 'bag', '.']
[Step d] : After removing punctuations from text
['use', 'everything', 'acoustic', 'bass', 'ukuleles', 'know', 'smaller', 'model', 'available', 'ukes', 'violins', 'etc', 'n',
't', 'yet', 'ordered', 'work', 'smaller', 'instruments', 'one', 'n', 't', 'extend', 'feet', 'maximum', 'width', 're', 'gentle',
'instruments', 'grippy', 'material', 'keeps', 'secure', 'greatest', 'benefit', 'writing', 'music', 'computer', 'needing', 'se
t', 'guitar', 'use', 'keyboard', 'mouse', 'easier', 'hanging', 'stand', 'several', 'gave', 'one', 'friend', 'christmas', 'wel
l', 've', 'used', 'mine', 'stage', 'folds', 'small', 'enough', 'fit', 'right', 'gig', 'bag', '']
[Step e] : After removing blank spaces from text
['use', 'everything', 'acoustic', 'bass', 'ukuleles', 'know', 'smaller', 'model', 'available', 'ukes', 'violins', 'etc', 'n',
't', 'yet', 'ordered', 'work', 'smaller', 'instruments', 'one', 'n', 't', 'extend', 'feet', 'maximum', 'width', 're', 'gentle',
'instruments', 'grippy', 'material', 'keeps', 'secure', 'greatest', 'benefit', 'writing', 'music', 'computer', 'needing', 'se
t', 'guitar', 'use', 'keyboard', 'mouse', 'easier', 'hanging', 'stand', 'several', 'gave', 'one', 'friend', 'christmas', 'wel
l', 've', 'used', 'mine', 'stage', 'folds', 'small', 'enough', 'fit', 'right', 'gig', 'bag']
File 3 after preprocessing:
use everything acoustic bass ukuleles know smaller model available ukes violins etc n t yet ordered work smaller instruments on
e n t extend feet maximum width re gentle instruments grippy material keeps secure greatest benefit writing music computer need
ing set guitar use keyboard mouse easier hanging stand several gave one friend christmas well ve used mine stage folds small en
ough fit right gig bag

```
File 4 before preprocessing:
Great price and good quality.  It didn't quite match the radius of my sound hole but it was close enough.

[Step a] : After Lower casing the text
great price and good quality.  it didn't quite match the radius of my sound hole but it was close enough.
[Step b] : After tokenizing the text
['great', 'price', 'and', 'good', 'quality', '.', 'it', 'did', "n't", 'quite', 'match', 'the', 'radius', 'of', 'my', 'sound',
'hole', 'but', 'it', 'was', 'close', 'enough', '.']
[Step c] : After removing the stopwords from the text
['great', 'price', 'good', 'quality', '.', "n't", 'quite', 'match', 'radius', 'sound', 'hole', 'close', 'enough', '.']
[Step d] : After removing punctuations from text
['great', 'price', 'good', 'quality', 'n', 't', 'quite', 'match', 'radius', 'sound', 'hole', 'close', 'enough', '']
[Step e] : After removing blank spaces from text
['great', 'price', 'good', 'quality', 'n', 't', 'quite', 'match', 'radius', 'sound', 'hole', 'close', 'enough']
File 4 after preprocessing:
great price good quality n t quite match radius sound hole close enough

File 5 before preprocessing:
I bought this bass to split time as my primary bass with my Dean Edge. This might be winning me over. The bass boost is outstan
ding. The active pickups really allow you to adjust to the sound you want. I recommend this for anyone. If you're a beginner  l
ike I was not too long ago, it's an excellent bass to start with. If you're on tour and/or music is making you money, this bass
will be beatiful on stage. The color is a bit darker than in the picture. But, all around, this is a great buy.

[Step a] : After Lower casing the text
i bought this bass to split time as my primary bass with my dean edge. this might be winning me over. the bass boost is outstan
ding. the active pickups really allow you to adjust to the sound you want. i recommend this for anyone. if you're a beginner  l
ike i was not too long ago, it's an excellent bass to start with. if you're on tour and/or music is making you money, this bass
will be beatiful on stage. the color is a bit darker than in the picture. but, all around, this is a great buy.
[Step b] : After tokenizing the text
['i', 'bought', 'this', 'bass', 'to', 'split', 'time', 'as', 'my', 'primary', 'bass', 'with', 'my', 'dean', 'edge', '.', 'thi
s', 'might', 'be', 'winning', 'me', 'over', '.', 'the', 'bass', 'boost', 'is', 'outstanding', '.', 'the', 'active', 'pickups',
'really', 'allow', 'you', 'to', 'adjust', 'to', 'the', 'sound', 'you', 'want', '.', 'i', 'recommend', 'this', 'for', 'anyone',
'.', 'if', 'you', "'re", 'a', 'beginner', 'like', 'i', 'was', 'not', 'too', 'long', 'ago', ',', 'it', "'s", 'an', 'excellent',
'bass', 'to', 'start', 'with', '.', 'if', 'you', "'re", 'on', 'tour', 'and/or', 'music', 'is', 'making', 'you', 'money', ',',
'this', 'bass', 'will', 'be', 'beatiful', 'on', 'stage', '.', 'the', 'color', 'is', 'a', 'bit', 'darker', 'than', 'in', 'the',
'picture', '.', 'but', ',', 'all', 'around', ',', 'this', 'is', 'a', 'great', 'buy', '.']
[Step c] : After removing the stopwords from the text
['bought', 'bass', 'split', 'time', 'primary', 'bass', 'dean', 'edge', '.', 'might', 'winning', '.', 'bass', 'boost', 'outstand
ing', '.', 'active', 'pickups', 'really', 'allow', 'adjust', 'sound', 'want', '.', 'recommend', 'anyone', '.', "'re", 'beginne
r', 'like', 'long', 'ago', ',', "'s", 'excellent', 'bass', 'start', '.', "'re", 'tour', 'and/or', 'music', 'making', 'money',
',', 'bass', 'beatiful', 'stage', '.', 'color', 'bit', 'darker', 'picture', '.', ',', 'around', ',', 'great', 'buy', '.']
[Step d] : After removing punctuations from text
['bought', 'bass', 'split', 'time', 'primary', 'bass', 'dean', 'edge', 'might', 'winning', 'bass', 'boost', 'outstanding', 'act
ive', 'pickups', 'really', 'allow', 'adjust', 'sound', 'want', 'recommend', 'anyone', 're', 'beginner', 'like', 'long', 'ago',
's', 'excellent', 'bass', 'start', 're', 'tour', 'and', 'or', 'music', 'making', 'money', 'bass', 'beatiful', 'stage', 'color',
'bit', 'darker', 'picture', 'around', 'great', 'buy', '']
[Step e] : After removing blank spaces from text
['bought', 'bass', 'split', 'time', 'primary', 'bass', 'dean', 'edge', 'might', 'winning', 'bass', 'boost', 'outstanding', 'act
ive', 'pickups', 'really', 'allow', 'adjust', 'sound', 'want', 'recommend', 'anyone', 're', 'beginner', 'like', 'long', 'ago',
's', 'excellent', 'bass', 'start', 're', 'tour', 'and', 'or', 'music', 'making', 'money', 'bass', 'beatiful', 'stage', 'color',
'bit', 'darker', 'picture', 'around', 'great', 'buy']
File 5 after preprocessing:
bought bass split time primary bass dean edge might winning bass boost outstanding active pickups really allow adjust sound wan
t recommend anyone re beginner like long ago s excellent bass start re tour and or music making money bass beatiful stage color
bit darker picture around great buy
```

2. Inverted Index

Function for preprocessing of user input

```
def preprocess(text):

    # Converting the text to lowercase.
    text = text.lower()
    # Performing tokenization
    tokens = word_tokenize(text)
    # Removing stopwords from the text
    stop_words = set(stopwords.words('english'))
    tokens = [w for w in tokens if not w in stop_words]

    # Removing punctuations
    #These lines commented below if uncommented will enable the data to be more information specific
    # This is because other kinds of punctuations are removed but not hyphen
#     split_tokens = []
#     for token in tokens:
#         if '=' in token or '/' in token:
#             split_tokens.extend(re.split(r'[=/]', token))
#         else:
#             split_tokens.append(token)

#     tokens = split_tokens

#     tokens = [word for word in tokens if word.isalpha() or '-' in word]
#     The below assumption separates all the words or terms in order to do punctuation

    tokens = re.split(r'[^a-zA-Z0-9]+', ' '.join(tokens))

    # Removing blank spaces in the tokens
    tokens = [w for w in tokens if w.strip()]

    # Removal of repeated tokens and preserving the order of tokens in the list
    tokens = list(dict.fromkeys(tokens))
    return tokens
```

Creating a dictionary, which stores a key and a value. Key are the terms and the value is a list having a list of the names of files in which the term is present and the second element of the list store the total frequency of the term or word in all the text files.

```
7]: #load the invertedindex data
    with open('inverted_index.pickle', 'rb') as handle:
        invertedindex = pickle.load(handle)

    def get_posting_list(term):
        if term in invertedindex:
            return invertedindex[term][0]
        else:
            return []

    def get_posting_listsize(term):
        if term in invertedindex:
            return invertedindex[term][1]
```

get_posting_list helps in getting the posting list and get_posting_listsize helps in getting the frequency of the posting list.

I have created for different functions and_operation for AND operation, or_operation for OR operation, or_not_operation for OR NOT and and_not_operation for AND NOT.

```
def and_operation(posting_list1,posting_list2,size_posting1,size_posting2):
    result = []
    i = 0
    j = 0
#     print("posting_list2:", posting_list2)
#     print("length of list1 ",len(posting_list1))
#     print("length of list2 ",len(posting_list2))
#     print("posting_list1:", posting_list1)
    while i < size_posting1 and j < size_posting2:
        if posting_list1[i] == posting_list2[j]:
            result.append(posting_list1[i])
            print("Updated result",result)
            print("posting_list1:", posting_list1[i])
            i += 1
            j += 1
        elif posting_list1[i] < posting_list2[j]:
            i += 1
        else:
            j += 1
    print(result)
    print("len(result)",len(result))
    return result
```

```
23]: def or_operation(posting_list1,posting_list2,size_posting1,size_posting2):
#     print("****** size_posting1 *****",type(size_posting1))
#     print("****** size_posting2 *****",size_posting2)
#     print(posting_list1)
#     print(posting_list2)
    or_result = []
    i,j = 0,0
    while i < size_posting1 and j < size_posting2:
        if posting_list1[i] == posting_list2[j]:
            or_result.append(posting_list1[i])
            i += 1
            j += 1
        elif posting_list1[i] > posting_list2[j]:
            or_result.append(posting_list2[j])
            j += 1
        else:
            or_result.append(posting_list1[i])
            i += 1

    if i < size_posting1:
        or_result.extend(posting_list1[i::])
    if j < size_posting2:
        or_result.extend(posting_list2[j::])

    return or_result
```

```
def or_not_operation(posting_list1,posting_list2,size_posting1,size_posting2):
#     print("size_posting2 : ",size_posting2)
#     print("size_posting1 : ",size_posting1)
#     print("posting_list2",posting_list2)
#     print("posting_list1",posting_list1)
    output = []

    diff = list(set(all_docs) - set(posting_list2))
#     print("length of diff",len(diff))
    output = or_operation(posting_list1,diff,size_posting1,len(diff))
#     print("the output is :",output)
    output = list(set(output))
#     print("length of output", len(output))
    return output
```

```
def and_not_operation(posting_list1, posting_list2, size_posting1, size_posting2):
    print("size_posting2 : ",size_posting2)
    print("size_posting1 : ",size_posting1)
    print("posting_list2",posting_list2)
    print("posting_list1",posting_list1)
    output = []

    diff = list(set(all_docs) - set(posting_list2))
#     print("length of diff",len(diff))
    output = and_operation(posting_list1,diff,size_posting1,len(diff))
#     print(diff)
#     print("the output is :",output)
    output = list(set(output))
#     print("length of output", len(output))
    return output
```

```
def main():
    n = int(input("Enter the number of queries you want to run:"))
    phrase = []
    operators = []
    for k in range(n):
        input_sequence = input("Enter the query phrase:")
        operations = input("Enter the operator (in a comma separated manner):")

        while len(word_tokenize(input_sequence)) != len(operations.split(',')) + 1:
            print("Number of tokens should be one more than the number of operators. Please re-enter.")
            input_sequence = input("Enter the query phrase:")
            operations = input("Enter the operator (in a comma-separated manner):")


        phrase.append(input_sequence)
        operators.append(operations)
#         print(operators)
    for i in range(n):
        query = phrase[i]
        query = query.lower()
        query = preprocess(query)
        op = operators[i]
        op = op.split(',')
        op = [ele.strip() for ele in op]
#         print("The query is", query)
        result = get_posting_list(query[0])
        size = get_posting_listsize(query[0])
        index = 0
#         print("query[index+1]",query[index+1])
        while index < len(op):
            if op[index] == 'AND':
                result = and_operation(result, get_posting_list(query[index+1]), size, get_posting_listsize(query[index+1]))
                size = len(result)
            elif op[index] == 'OR':
                result = or_operation(result, get_posting_list(query[index+1]), size, get_posting_listsize(query[index+1]))
```

Main function processes the input query. First user is asked for n , number of queries as input. Then user is asked to give query and

respective operations. I check that query is not more than the operations and vice -versa.

3. I have created a dictionary which stores terms or words as key and it's value is a dictionary containing file names as keys and the number of times the word occurs in it as frequency (value).

In [3]:
```
def create_dict(tokens, filename, db):
    # Iterating over every token in the list of tokens
    for i, token in enumerate(tokens):
        if token in db:
            if filename in db[token]:
                db[token][filename].append(i)
            else:
                db[token][filename] = [i]
        else:
            db[token] = {filename: [i]}
```

In [4]:
```
def build_index(directory):
    os.chdir(directory)
    db = {}
    for filename in tqdm(os.listdir()):
        with open(filename, 'r') as f:
            text = f.read()
        tokens = preprocess(text)
        create_dict(tokens, filename, db)
    return db
```

In [5]:
```
db = build_index('C:/Users/JAYSA/Downloads/IR/modified_preprocessed_files')
```

100% 999/999 [00:10<00:00, 100.79it/s]