T. Satya Himavanth

Y20CS179

ASSIGNMENT TEST - I

1) (a) Explain how to build an inverted index.

Ans:- Steps to be followed to build an inverted index are:-

Step 1:- Collect the documents to be indexed:

Ex:- Doc1, Doc2, Doc3 ...

Step 2:- Tokenize the text, turning each document into a list of tokens.

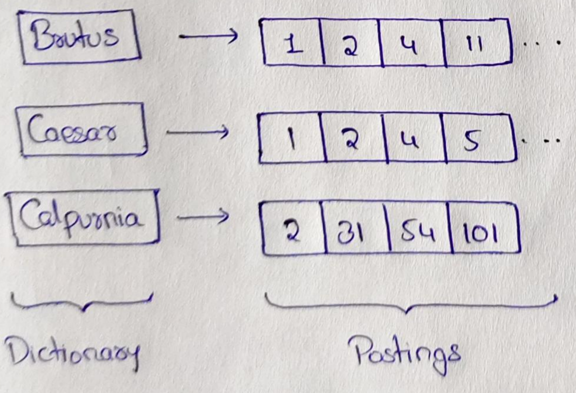
Ex:- Tokens in Doc1:- Friends, Romans, countrymen, ...

Step 3:- Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms.

Ex:- friend, roman, countryman, ...

Step 4:- Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and posting.

Ex:-



* Indexing terms should be in Lexicographical order.
* Postings / DocID's should be in ascending order.

(b) Write algorithms for processing Boolean queries.

Ans: S = list of all docid.

```
def intersect (lst1, lst2):
    lst3 = [val for val in lst1 if val in lst2]
    return lst3.

def Union (lst1, lst2):
    lst3 = ~~val~~ list (set (lst1) | set (lst2))
    return lst3

def Not (lst1, lst2):
    s = set ( lst2)
    lst3 = [val for val in lst1 if x not in s]
    return lst 3

Alg Boolean (query):
    q = query.split()
    i=0
    a= dictionary [q[i]]
    while (i < len(q) -1):
        i+=1
        con =q[i]
        i+=1
        b= dictionary [q[i]]
        if con=='and':
            a = intersect (a,b)
        elif con=='or':
            b = Union (a,b)
        elif con== 'notand' or con== 'andnot':
            a = intersect (a, Not (s,b))
    elif  con=='notor' or con=='or not'
        a= Union (a, Not (s,b))
return a.
```

Ex: "a and are"

dictionary ['a'] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

dictionary ['are'] = [5, 7, 8, 9, 10]

Output: [5, 7, 8, 9, 10]

(2) Explain the process of determining vocabulary of terms.

Ans: Process of determining vocabulary of terms is as follows:

1) Tokenization:

It is a process of removing punctuations and breaking the documents into words. These words are called tokens. In some cases tokens are further reduced by removing some characters. During the process of tokenization there are certain issues due to ~~the~~ different types of writing etc.

i) Term: O'Neill , aren't

→ O, neill does't make sense as it is a name .

→ aren , t also does't make sense at it a singl word.

So in some cases tokenization is not possible.

ii) Language identification:

Since tokenization is language specific, it thus requires the language of document to be known. Language identification based on classifiers that use short character sequences as features is highly effective.

iii) Hyphens:

In English hyphenation is used for various purposes ranging from spliting up vowels in words (co-education) to joining nouns as names (Hewlett-Packard) to a copyediting device to show word grouping (the hold-him-back).

We need to identify the type of word before tokenizing it. Not only for words but also for numbers ((800) 234-2333) and also date (Mar 11, 1983) etc.

* Similarly others - Compounds, Compound-splitter, word segmentation.

2) Stop words:

Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words. The general strategy for determining a stop list is to sort the terms by collection frequency and then to take the most frequent terms, often hard-filtered for their semantic content relative to the domain of the documents being indexed, as a stop list, the members of which are then discarded during indexing.

Some times removing stop words might change the output the search.

Ex: "President of the United States".

→ "President" and "United States".

3) Normalization:

Token normalization is the process of canonicalizing tokens so that matches occur despite superficial differences in the character sequences of the tokens.

The most standard way to normalize is to implicitly create equivalence classes, which are normally named after one-member of the set.

Ex: class: antidiscriminatory

→ anti-discriminatory

→ antidiscriminatory.

An alternative is to create equivalence classes that maintain relations between unnormalized tokens.

Ex: class: vehical

→ car

→ automobile

⊛ Other ways are : Capitalization / case-folding.

Accents & diacritics.

4) Stemming and lemmatization:-

"The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Ex:-

am, are, is ⇒ be

car, cars, car's, cars' ⇒ car

i) Stemming:- It is a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time and often includes the removal of derivational affixes.

ii) Lemmatization:- It usually refers to doing things properly with the use of vocablary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as lemma.

Ex:- word = saw

stemming :- s

Lemmatization :- see (or) saw

⊛ Algorithm for stemming is "Porter Stemmer"

⊛ For Lemmatization we use Natural Language processing which does full morphological analysis to accurately identify the lemma.