CSE225 Data Structures, 2018 (FALL)

PROJECT #1

Deadline: 30.10.2018, 24:00

Demos: 31.10.2018 and 1.11.2018

Text representation with bag-of-words (BOW) approach

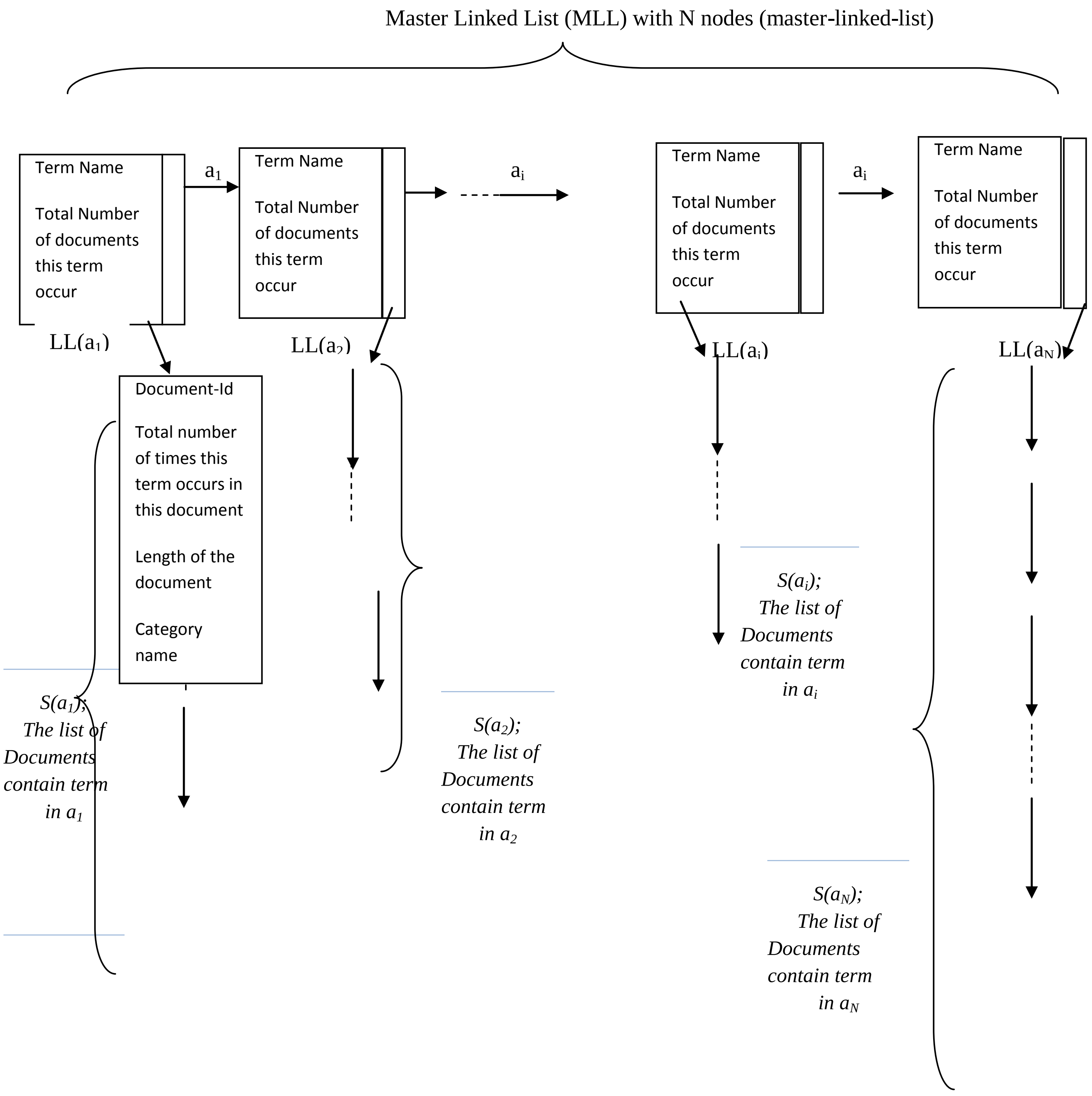
Traditional representation methodology of documents in the literature is called Bag of Words (BOW) feature representation. This representation symbolizes the terms and their corresponding frequencies in a document and it is also named as Vector Space Model (VSM). Each of these terms in the same document represents an independent dimension in a vector space [1]. The order of words in the sentences is completely lost in bag representation like in sets. This approach mainly emphasizes the frequency of terms. The BOW methodology makes the representation of words simpler in documents; still; it has several problems. One of them is sparse vector representation. This makes the computation expensive especially for real world scenarios, which include big data in textual domains. To address this problem; this project aims to create more a more efficient representation of BOW model by using Linked-Lists. Consequently, this project is a programming assignment in C, which aims to build an algorithm based on linked-lists that will build an efficient representation of documents.

Your program needs to open and read text files under the following directories: *sport, magazine and health*. These are 3 categories of 1150Haber dataset [2]. The number of documents in these categories will be arbitrary. Furthermore, the number of terms in these documents will also be arbitrary. In other words, the length of these files will be arbitrary.

Your program is expected to do the followings:

a) (40 points) You need to read all the documents under all the categories. Then you need to build a Master Linked-List (MLL). Each node in this MLL needs to represent a different term in these documents. All the terms in these documents are expected to be in the MLL. There will be cases, the same word occur in different documents, or in the same document. Then, you do not need to add a term into the MLL if it already exists. In other words, be careful about not entering the duplicate records into the MLL. This list needs to be in ascending order.

   Each record in MLL has 2 pointers: The first of them is for the next record in MLL. The second of them is for the starting record of another Linked-List. This sub Linked-List will represent the documents that contain the term in this record of MLL.
   Figure-1 shows the structure of MLL.



Master Linked List (MLL) with N nodes (master-linked-list)
Term Name
Total Number of documents this term occur
a_1
Term Name
Total Number of documents this term occur
a_i
Term Name
Total Number of documents this term occur
a_i
Term Name
Total Number of documents this term occur
$LL(a_1)$
$LL(a_2)$
$LL(a_i)$
$LL(a_N)$
Document-Id
Total number of times this term occurs in this document
Length of the document
Category name
$S(a_1)$; The list of Documents contain term in a_1
$S(a_2)$; The list of Documents contain term in a_2
$S(a_i)$; The list of Documents contain term in a_i
$S(a_N)$; The list of Documents contain term in a_N

Figure.1. Schema of the Master-Linked-List and the Linked-List of Documents

b) (30 points) Finding stop words/general words (noise in the documents): You need to find the first 5 general words which occur in all of the categories. In other words, you need to list the terms that are common for all of the categories. The output will be like the following (it needs to be listed in ascending order):

Term-1: aaa
Term-2: bbb
Term-3: ccc
Term-4: ddd
Term-5: eee

c) (30 points) Finding discriminating words: You need to find the first 5 words for each category which occur in that category only, not in other categories. The output will be like the following: (it needs to be listed in ascending order)

Category-1	Category-2	Category-3
Term-1	Term-1a	Term-1b
Term-2	Term-2a	Term-2b
Term-3	Term-3a	Term-3b
Term-4	Term-4a	Term-4b
Term-5	Term-5a	Term-5b

Important Notes:

In your demo, we will run your program by **reading arbitrary-length input files. You need to store these files into linked-lists.**

In demo, we will use different input files and we will check if your program find the correct results or not. We will also check your data structure your design architecture.
Of course, other questions based on your implementation and coding structure will be asked you during your demo. These questions will be those kinds of questions that could be answered by only the students who really implement his/her project by himself/herself.

The main goal of this project is to be familiar with linked-list. Therefore, if you use arrays instead of linked-lists then you will get zero, unfortunately.

You are responsible for demonstrating your program to your TA Berna Altınel on the scheduled day that will be announced later.

CODE SUBMISSION:

You should use the following email address in order to submit your code:

cse225.marmara.2018 at gmail dot com

Your any submission after the project submission due date, will not taken into consideration.

You are required to exhibit an ***individual effort*** on this project. Any potential violation of this rule will lead everyone involved to **failing from the course** and necessary disciplinary actions will be taken.
Good luck!!!

REFERENCES

[1] Salton, G., Yang, C.S., 1973. On the Specification of Term Values in Automatic Indexing, Journal of Documentation, 29(4):11-21.

[2] Amasyalı, M. F. and Beken, A. Türkçe Kelimelerin Anlamsal Benzerliklerinin Ölçülmesi ve Metin Siniflandirmada Kullanilmasi, in Proc IEEE Sinyal İşleme ve İletişim Uygulamalari Kurultayi (SIU), IEEE Press, 2009.