CSE225 Data Structures, 2019 (FALL)

PROJECT #1 / Deadline: 5.11.2019

Text representation with Higher-Order Paths

Advantages of using higher-order paths [1] between documents are illustrated in Fig. 1. In this figure, there are three documents, d_1, d_2 and d_3 which include a set of terms $\{t_1, t_2, t_3\}$, $\{t_3, t_4, t_5\}$ and $\{t_4, t_5\}$ respectively. Using a traditional similarity measure which is based on the shared terms (e.g. dot product), similarity value between documents d_1 and d_3 will be zero since they do not share any terms. But in fact these two documents have some similarities in the context of the dataset through d_2 as it can be seen in Fig. 1. This supports the idea that using higher-order paths between documents, it is possible to obtain a non-zero similarity value between d_1 and d_3 which was not possible in traditional Bag of Words (BOW) [2] representation. This value becomes larger if there are many interconnecting documents like d_2 between d_1 and d_3. This may stem from the reason that the two documents are written on the same topic using two different but semantically closer sets of terms.

This project aims to represent these higher-order paths 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.

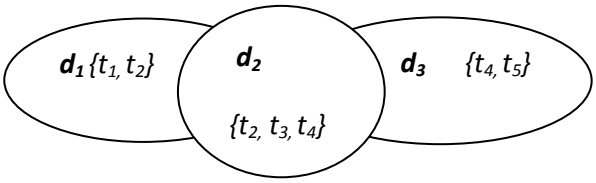
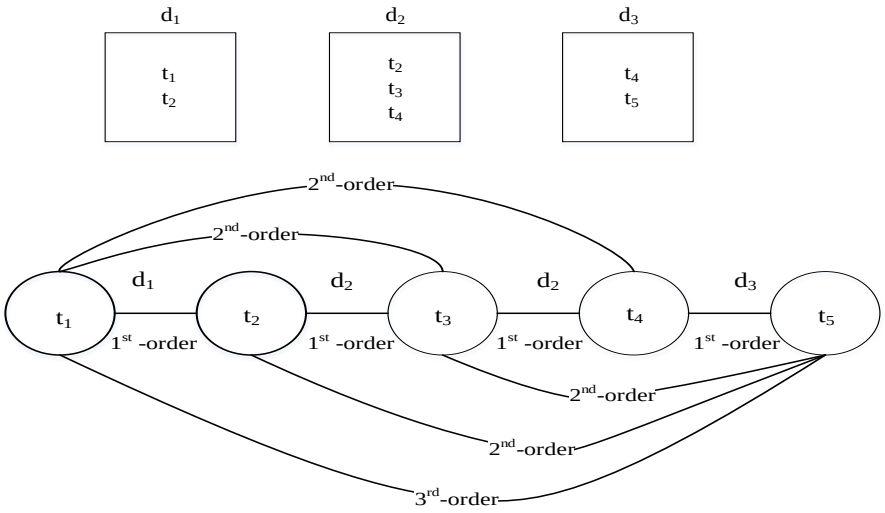


Fig. 1. a) Illustration of higher-order paths



1^{st}-order term co-occurrence $\{t_1, t_2\}$, $\{t_2, t_3\}$, $\{t_3, t_4\}$, $\{t_2, t_4\}$, $\{t_4, t_5\}$
2^{nd}-order term co-occurrence $\{t_1, t_3\}$, $\{t_1, t_4\}$, $\{t_2, t_5\}$, $\{t_3, t_5\}$
3^{rd}-order term co-occurrence $\{t_1, t_5\}$

Fig. 2. b) Graphical demonstration of first-order, second-order and third-order paths between terms through 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 [3]. 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) (60 points) You need to read all the documents under all the categories. Then you need to build a 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. After reading and storing all your data into Linked list, you are expected to find 1^{st}, 2^{nd} and 3^{rd} order term co-occurrences as shown below.

1^{st}-order term co-occurrence $\{t_1, t_2\}$, $\{t_2, t_3\}$, $\{t_3, t_4\}$, $\{t_2, t_4\}$, $\{t_4, t_5\}$
2^{nd}-order term co-occurrence $\{t_1, t_3\}$, $\{t_1, t_4\}$, $\{t_2, t_5\}$, $\{t_3, t_5\}$
3^{rd}-order term co-occurrence $\{t_1, t_5\}$

There are several ways to build a suitable data structure by using Linked-List in order to find higher-order paths. ***In this project, you are also expected to write your algorithm's analysis (i.e. Give the big-O time (order) of your code).***

Output:
1^{st}-order term co-occurrence $\{t_1, t_2\}$, $\{t_2, t_3\}$, $\{t_3, t_4\}$, $\{t_2, t_4\}$, $\{t_4, t_5\}$
2^{nd}-order term co-occurrence $\{t_1, t_3\}$, $\{t_1, t_4\}$, $\{t_2, t_5\}$, $\{t_3, t_5\}$
3^{rd}-order term co-occurrence $\{t_1, t_5\}$

and

big-O time (order) of your code

b) (20 points)
Output: Your program will output following information:
Most frequent 10 words in the input set of documents, sorted descending by their term frequency (tf) coupled with their tf values (comma seperated format, example: car;7)

c) (20 points)
Output: Most frequent 10 words in the input set of documents, sorted descending by their term frequency*inverse document frequency (idf) coupled with their tf-idf values (comma seperated format, example: car;2.8)

$IDF(t) = \log_e N/n$

N: Total number of documents
n: Number of documents with term t in it.

Important Notes:

After the deadline; we will collect all codes and run our plagiarism-check program as we discussed in class.
You are also expected to submit 1 page of your report which includes answers of the above questions.

We may 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.

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

CODE SUBMISSION:

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

cse225.marmara.2019 at gmail dot com

REPORT SUBMISSION:

Until the end of PS on 06.11.2019.

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] Altinel, B., Ganiz, M. C., & Diri, B., 2014 . A semantic kernel for text classification based on iterative higher–order relations between words and documents. In *International Conference on Artificial Intelligence and Soft Computing* (pp. 505-517). Springer, Cham.

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

[3] 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.