**CMPE414 - Information Retrieval**

**Project 2: Document Vectors and Index Inversion.**

**Due Date:** 26/04/2019

**Objectives:** Have practical Knowledge on Inverted Indexes

Have experience with a workable structure.

Create/Use/Understand a data format that could be useful in further pursuits.

Prepare Data for the next project

**Project:** In this project, your task is as follows:

You are given a dataset that is called "radikal", which contains the contents of the newspaper Radikal articles. When you open the dataset, you will see that it has a html-like structure. The header of each article contains the hmtl name of the article, followed by its genre. (Such as anahaber, or sports). The associated files with the dataset can be found in the below link (I have also added a processed version of the file to save you some trouble):

https://www.dropbox.com/s/6mi69k9gj9cg4jw/radikal.corpus?dl=0

https://www.dropbox.com/s/3mfcff1nss3qm70/radikal.corpus\_preprocessed?dl=0

In this project you have two simple tasks, and for those tasks you may need to use the Trie structure from the first project (It is up to you though).

1 - CREATE A DOCUMENT VECTOR COLLECTION:

In this collection our aim is to:

- represent each article (document) as a vector of terms

- at the same time convert the text in the article into numeric IDs (numerating the documents in the final product)

- and at the same time we want to store all information in a lossless manner in a set of files.

For this purpose, we will define 4 new files. Below are these files' contents and descriptions:

**.Info File:** contains information about the total number of documents, terms and etc. You can use the information here to precisely access to other files.

The info file will contain two lines. These are:

numberOfDocuments <X>

numberOfUniqueTerms <Y> , where X and Y are the number of documents and distinct terms in the collection.

**.terms File:** contains every distinct word as a separate line. Each word text is accompanied by 3 other numbers: <the ID of the term, the number of documents that contain the term, the total number of occurrences in the dataset>. You can use the line as the ID number of the term (this is provided as a simple workaround). Finally, the fields in the terms file should be tab separated.

**.docs File;** which contains <document name, document genre, the total number of terms in the document, the number of distinct terms in the document >. name and genre of a document will be two text fields and all fields in the file will again be tab separated.

**.DV (Document Vector) File:** for each document in the collection, .DV file maintains a line which contains information on the respective document on the .docs file. Each line of the .DV file contains a set of integer tupleS <Term\_ID, Term\_Appearance\_Count> pairs. For example the line "1 1 2 1 3 1" in the .DV file means the document contains terms 1, 2, and 3; each appearing 1 time. Obviously all numbers in this file is also tab separated. (you are advised to also write the term id's in sorted order !!!!, for the next projects sake)

2 - CREATE AN INVERTED INDEX

In this project, you are also asked to create an inverted index from this file. That is, you are asked to create 4 files: on top of the previous .info, .terms, .docs files, you are asked to produce an .IDV (Inverted Document Vector) file, which maintains a line for each of the terms in the similar manner as the .DV file. (That is instead of the Term\_ID values, we will have Doc\_Id values in the IDV file. Also, we wont be having anything like a Term\_Appearance\_Count value, in the IDV file. So, for a term the term posting list will be <doc\_ID1 doc\_ID2 ....> separated with a tab character )

HINTS: you are most likely to go over the corpus file multiple files in this project. For example in the first part, easiest thing to do is:

- go over the file once to retrieve the dictionary, while counting the total term counts,

- go over the file a second time to convert each article into its DV form. (getting the document terms in an array and sorting them may be a nice idea here, although completely optional)

And for the second part of the project, you may check out section 5 of the following paper, which describe a meriad of algorithms for index inversion. (However note that since the file we are working on is small, you may most probably want something that does not contain merging)

Paper Link:

https://www.dropbox.com/s/18h6e9a9ew2zsnb/inverted%20files%20for%20text%20search%20engines.pdf?dl=0

Good Luck and Have Fun!