HO CHI MINH CITY UNIVERSITY OF SCIENCE
ADVANCED PROGRAM IN COMPUTER SCIENCE

CS419: IR – Final project

Luu Duc Huy	1151019
Nguyen Duc Thinh	1151034

Table of Contents

Introduction to the project

The topic our team chose is Text retrieval.

The C# program helps user search for documents that are relevant to a given query.

To use the program, use inputs a query to the small text box. Then press the Search button. The program will return a list names of relevant documents.

Development environment

The program was written in C#.

The program was developed and built with Visual Studio 2013.

External libraries

In this project we use an external library: MathNet. It can be install via Visual Studio NuGet manager. If you opened the program by opening the submitted project files, no more installation is needed. If you want to install MathNet for a new project, the steps are:

- Right click on the project in Solution Explorer window
- Choose Manage NuGet packages
- Search for Math.NET
- Install "Math.NET Numerics".
- Click Close.

Project structure

This program has 3 main classes: Program, Document and Corpus.

Program

The main class any other Windows Form project would have.

Document

Document class represent document in real life. It has *docname* to store the name of the document. In this project, we take file name as document name.

Wordnumber stores the number of distinct words in that document.

wordlist is a map that match each distinct word in the document with the times it appears in that document.

Another attribute is *vector*. This will represent the concept vector of the document.

Corpus

Corpus is a group of documents. In *Corpus* class, we have several attributes.
documents: this is a map that match each document name with an instant of *Document* class that represent that document.
vectorspace: the vector of the corpus. It will be explained in the Algorithm section.
savefilepath: an array of path to save other files.
masterdictionary: a map that match all the "selected" word to its number of occurrence in the whole corpus. More detail in Algorithm section.
Note: other attributes weren't mentioned here are some orphan attributes left in the source code after previous assignments.

Other files

stopwords_vn.txt: stores all the stopwords.
vpresult.txt: store the "selected" words. More detail in Algorithm section.

Algorithm

The program has 2 modes. One is Build mode and the other is Search mode.

Build mode

In the build mode, the program will generate a lot of binary files in folder *vectordata.* These files are the concept vectors of documents. Each document has one concept vector store in the file named after the document file name. For example, for document file *DTR_050330_1823.txt*, the program will generate a binary file named *DTR_050330_1823.txt.bin* to store the vector of that document.

First, the program create a corpus for the collection. Then the program scan through all the documents. After reading all the word in each document, it create an instant of Document to represent that document. After this phase, the *vector* attribute is still null.

While scanning documents, the program skips the words in the stopword list

When all documents were scanned, the *masterdictionary* in Corpus stores all the distinct words in the collection and their frequency of occurrence. Now, the program will eliminate all the words that appear too many time or too little time in the collections. These words don't contribute much in the retrieval process.

From the remained words, the program chooses randomly 2500 words. These 2500 words will be store in *vectorspace*. They are also stored in *vpresult.txt* file.

From these 2500 words, the program build concept vector for each document and save those vectors to binary files.

Build mode takes a lot of time (about 1.5 hour). Thus it should be executed once only. There already are binary files in the submitted files.

Search mode

The program load concept vectors from binary files.

The query is represented as a document. The raw query user inputted will be proceeded as a document: remove stop words, words that appear so many time or so little time. Then a concept vector for that query is created based on the loaded data from binary files.

After that, the program calculate similarity of the query with collection of documents by compute cosine of pairs of vectors. If the result is greater than 0.5, the document will be added to the result list and display to screen.

Used technique

Here are some techniques we used in the project:

- Tf-idf
- Inverted index
- Removing stop words
- Normalize word
- Tokenization
- Use Zipf law to remove “important” words
- Use database to save time when searching in large dataset.

Performance

Time to build database with 2500 words selected randomly: less than 2 hours.

Time to search for a query that has ~1,000 character: 8 minute

In search mode, the program takes 1,500MB of memory.