Hacettepe University
Computer Science and Engineering Department

Advisers	:	RA Ali ÇAĞLAYAN alicaglayan@cs.hacettepe.edu.tr
	:	Dr. Erkut ERDEM, Dr. Mustafa EGE
Course	:	BBM204 Programming Lab. II
Experiment No	:	4
Subject	:	Graphs, Spanning Trees
Programming Language	:	Java
Deadline	:	23.05.2014 23:59

1 Introduction

Graphs are one of the most important data structures and have been used in a wide range of fields in computer science. In this experiment, you will develop an application on one of these fields which is Natural Language Processing. By the help of this experiment, you will have chance to practice on Graphs, Spanning trees and learn about some basic concepts of Natural Language Processing.

The spread of World Wide Web and availability of electronic documents have increased the importance of automatic document classification to organize information. Classification is a field of Machine Learning and aims to separate a given group of data sets into groups in which the same group members are similar to each other and dissimilar to the other group members. Classification can be done by using supervised or unsupervised methods. Supervised methods use labeled examples for training whereas in unsupervised methods (which are mostly known as clustering) no labeled examples are provided for training.

In general, a vector model is used to represent a document for clustering. In this model, a vector keeps terms/words that may appear in a document. This is simple way which considers the numbers of terms or presence/absence of terms and ignores the positions of terms in a document. After the representation of documents, similarity between documents is going to be computed. For that, Cosine distance is used in general.

Spanning trees are subgraphs of given weighted graphs in which they have no cycles and contain minimal set of edges that connect all vertices. Spanning trees can be used in various applications and one of these applications is clustering which you are going to use in this assignment.

2 Experiment

This section contains four subsections. The first one is problem definition, and the second one describes the content of your reports. The third one summarizes the **constraints** that you should obey (Otherwise your experiment will not be evaluated by your adviser). The last one is about submission.

2.1 Problem Definition

In this experiment, you are supposed to develop a simple document clustering system. You are responsible for using Graphs and Spanning Trees in Java programming language. The system will process several document files and generate results of clustering to an output file. You will measure similarity between documents by using terms which is given in a dictionary file.

Document similarity or relatedness is a concept that measure how similar two documents are. For example, *Live at Wembley '86* and *The Wall – Live in Berlin* (Figure 1) Wikipedia documents are similar and related to each other, whereas *Life Is Beautiful* (Figure 2) is not similar to these documents.

In Figure 3, a graph which is constructed by the Wikipedia page documents is given. You can see that semantically similar documents are located close to each other on the graph, whereas different documents are distant from each other.

You are supposed to use the *bag-of-words model* for each document in this experiment. In this model, each document is represented as a bag (vector) of words and each entry of the bag refers to count of the corresponding term in the dictionary. You will use a dictionary (term dictionary file) to build a vector for each document. Each word in the term dictionary is an entry of the vectors. Remember that, terms must be handled by ignoring case. Once you construct vectors for each document, you are going to compute the similarity between these vectors.

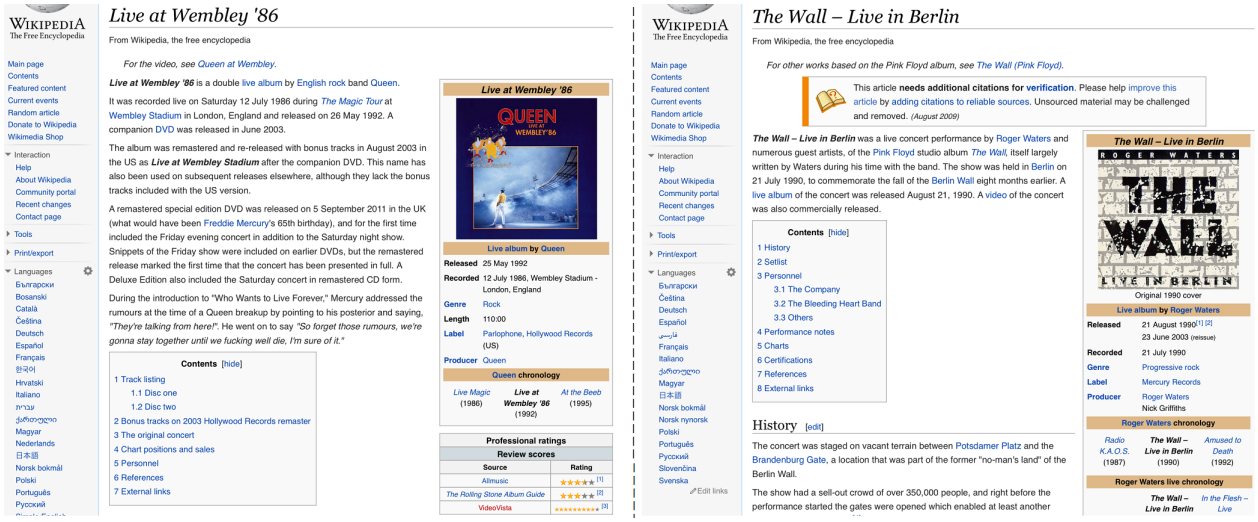

Figure 1: *Live at Wembley '86* and *The Wall – Live in Berlin* Wikipedia pages [1, 2].

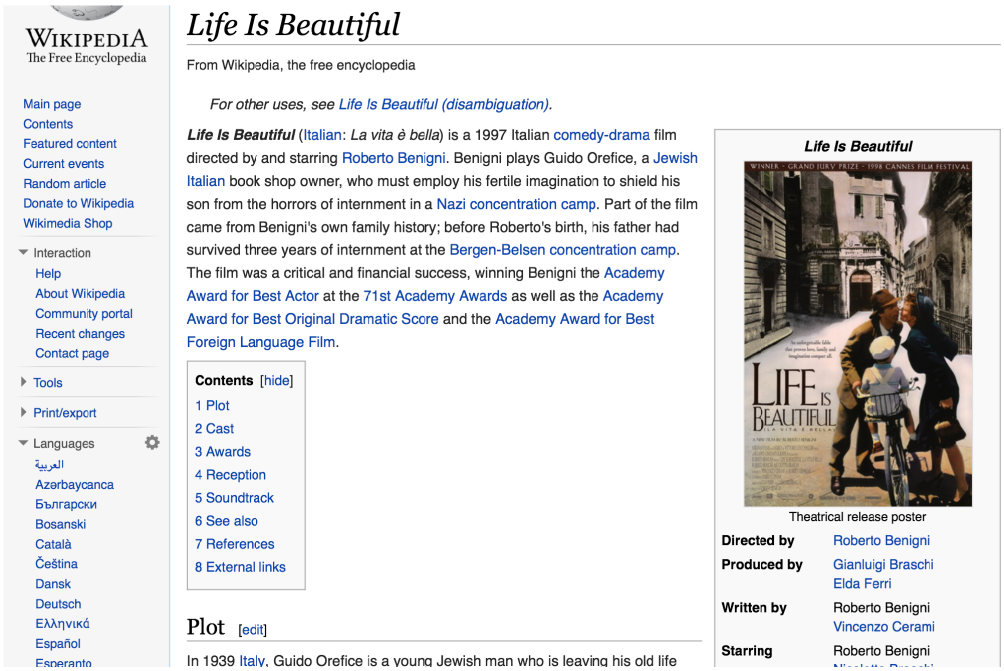

Figure 2: Italian film *Life Is Beautiful* Wikipedia page [3].

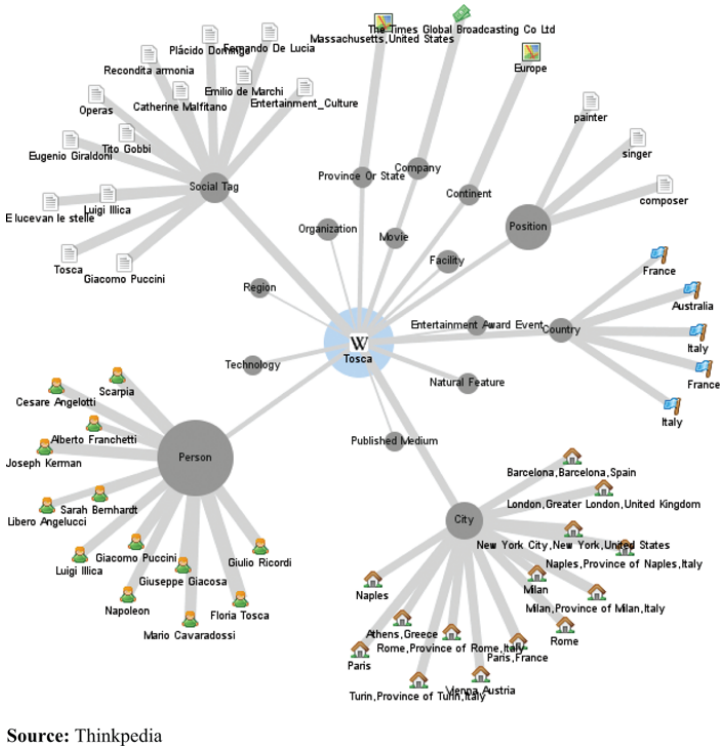

Figure 3: A sample semantic clusters of Wikipedia documents [5].

You will use *Cosine similarity* to measure the semantic similarity between vectors which represent the documents. Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle [6]. Cosine similarity is based on magnitude of values and negative space is not used. Thus, the similarity range is between 0 and 1. The bigger the value, the more similar documents. Cosine similarity is calculated as follows:

$$sim(v_1, v_2) = \cos(\theta) = \frac{v_1.v_2}{\|v_1\|\|v_2\|} = \frac{\sum_{i=1}^{n} v_{1i} \times v_{2i}}{\sqrt{\sum_{i=1}^{n} {v_{1i}}^2} \times \sqrt{\sum_{i=1}^{n} {v_{2i}}^2}} \tag{1}$$

In the part of clustering the documents, you will use spanning trees on your graph which you constructed according to the cosine similarity before. Spanning tree is a common way for clustering data. Once a spanning tree on a given graph is constructed, k clusters can be obtained by removing $k-1$ edges that have the minimum/maximum weights on the spanning tree. Thus, k semantically similar clusters are obtained. In our case, each document or document representation (vector) is a vertex and the similarity between these documents represents edges between vertices. Therefore, first you are going to compute the document vectors and then calculate the similarities between each vector pair to construct a weighted undirected graph. Finally, you will use the weighted graph to find the spanning tree and cut $k-1$ edges to obtain k clusters. The number of clusters must be taken as an argument parameter in your program.

2.1.1 The Way of Execution

The program will be executed with four command line arguments:

<dictionary file> <the file including documents' names> <number of clusters> <output file>

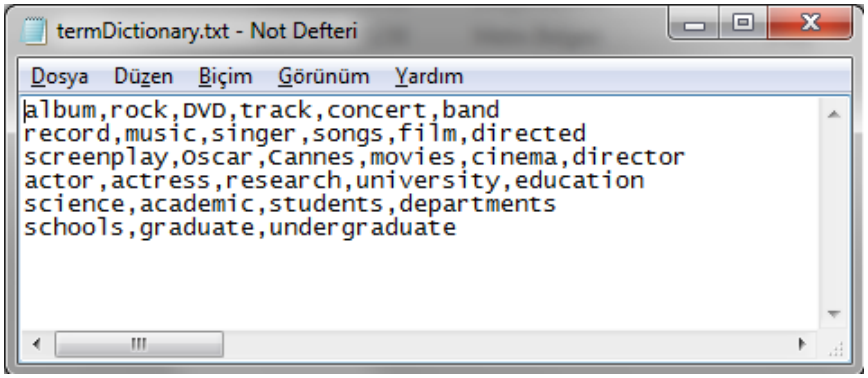

Figure 4: A sample term dictionary file.

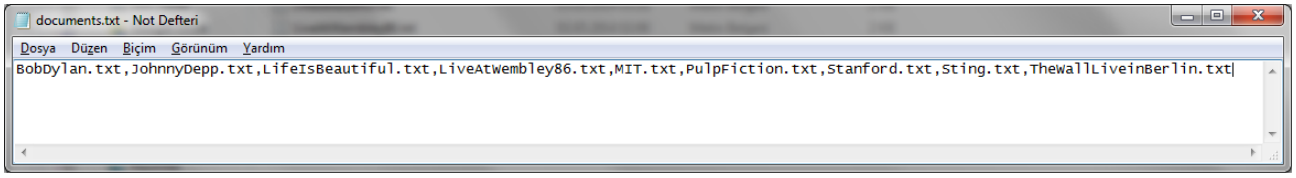

Figure 5: A sample documents file which includes the names of all the documents.

Usage example:
>javac Main.java
>java Main termDictionary.txt documents.txt 3 output.txt

There are three types of data input files and one output file. Except the names of documents which will be classified, all the file names will be taken as program arguments. The names of documents will be given in the second argument file (documents.txt in the example). The format of each file type is given below.

2.1.2 Term Dictionary File

This file includes common terms of documents. You are going to use this file to compute vectors for each document. Therefore, each vector size is equal and the number of terms inside this file gives vector length. Each term is separated by a comma. The format is as follows:

$<term_1>$ <comma> $<term_2>$ <comma> $<term_3>$ ... $<term_n>$

A sample term dictionary file is shown in Figure 4.

2.1.3 Documents File

This file includes the name of the documents which are going to classify. Each document name is separated by a comma. The format is as follows:

$<document_1>$ <comma> $<document_2>$ <comma> $<document_3>$ ... $<document_n>$
A sample documents file is shown in Figure 5.

2.1.4 A Document File

A document file is about a subject which can vary and includes explanation about the subject with multiple lines. The sample documents which are given in this assignment, are copies of manipulated Wikipedia English page related to the subjects. There is not a regular format of a document file. Unlike other files, all the document files will be taken from a directory named **"documents"**.

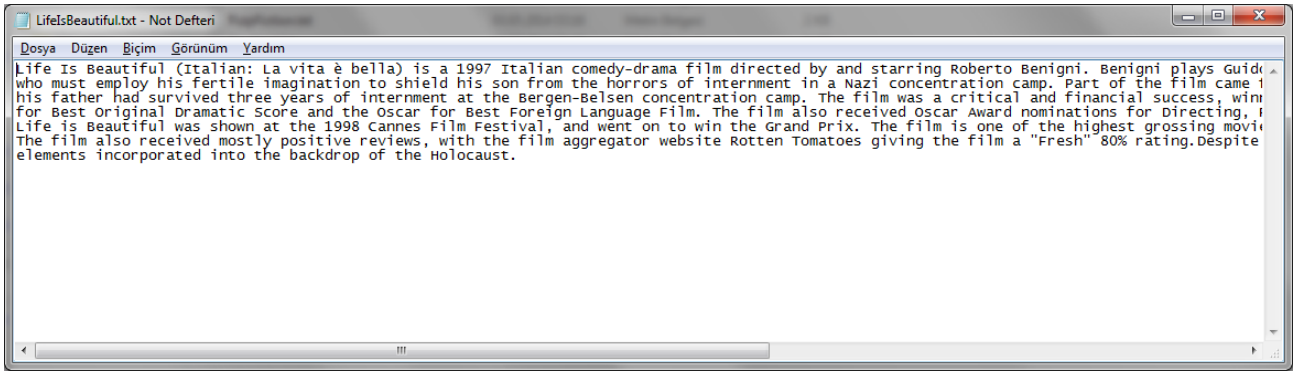

Figure 6: A sample document file. This document is about Life is Beautiful (Italian: La vita è bella) movie. The document is based on Wikipedia page [3] and some text is manipulated for our example.

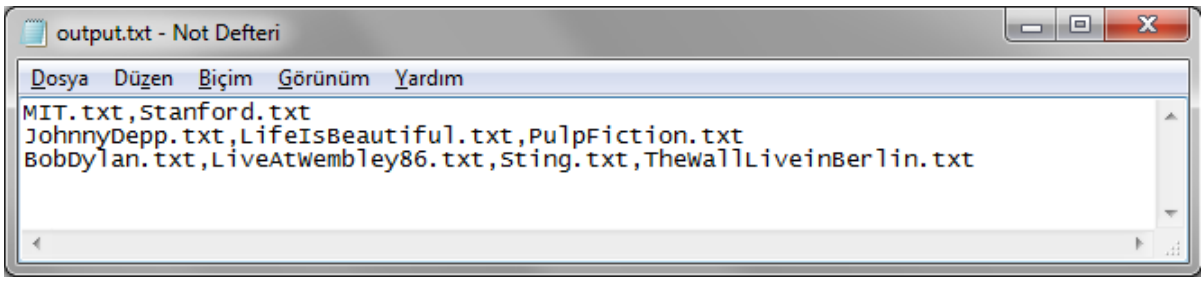

Figure 7: A Sample output file.

Therefore, you should remember that a document file name which you read before from documents file doesn't give the file path. You should add **/documents** in front of the name for path (e.g. */documents/LifeIsBeautiful.txt*).

A sample document file is shown in Figure 6.

2.1.5 Output File

The result of classification will be printed to the specified output file. Contents of each cluster will be written to this file. Cluster members will be written in alphabetical order (in increasing order) and clusters will be written according to the number of members in each cluster in increasing order (i.e. cluster that has the minimum number of members will be written first). Cluster members must be delimited by commas.

According to above definitions a sample output file is given in Figure 7.

Further examples which give more details will be provided at the course's FTP site and via Piazza.

2.2 Report

The structure of report is described below:

- **Cover Page**
- **Problem**, describe the problem with your own sentences and give a review of document classification in literature.
- **Solution**, describe details of your solution, stating its advantages and disadvantages technically. Give your algorithms and execution flow between subprograms.
- **References**, give the references you used throughout your work at the end of your report.

2.3 Constraints

You should obey these constraints. Otherwise your experiment will not be evaluated properly.

1. The methods' and attributes' names should be satisfied the most common naming conventions in Java.
2. You should model entities of the system with classes.
3. You are responsible for a correct model design of your program. Your design should be accurate.
4. You should use Graphs and Spanning Trees.
5. All the input files and output file will be taken as command line arguments.
6. **Take care about the output file format explained above. Your experiments will be evaluated automatically, so your output files should be comply exactly the same with the advisor's one.**

2.4 Submission

The submissions will be accepted only by `http://submit.cs.hacettepe.edu.tr`. The submit should be done until 23/05/2014 23:59 for all sections. Submission format is as follows:

<**student_id**>**.zip**(example: 21521512.zip)
[**src**]
–Main.java
–*.java
–*.zip
[**report**]
–report.pdf

3 Grading

Your Experiment results will be evaluated with the following rules. They are fixed. They will never be changed. Hence obey these rules. Before you contest your grade re-evaluate yourself with these rules.

- The execution of program have 75 points.
- The submit format have 1 points.
- The report have 24 points: 1 point for **Cover Page**, 8 points for **Problem**, 14 points for **Solution** and 1 point for **References**. Your final report grade will be calculated as $(Execution * Report)/75$ and added to your execution grade for final score.

4 Notes

These notes are very important please read them and obey them.

- **No submissions except Submission System of Computer Engineering Department will be accepted. Do not send your experiment with emails.**
- **Do not miss the deadline. No submissions will be accepted after 23.05.2014 23:59.**
- Respect your adviser office hours: Monday, 13:00-16:00 (RA A. ÇAĞLAYAN).
- For every question or problem please contact with your adviser RA Ali ÇAĞLAYAN. Do not contact with teachers of experiment directly.

- You can send email to your adviser's email alicaglayan@cs.hacettepe.edu.tr.
- **You can ask your questions via Piazza (`https://piazza.com/hacettepe.edu.tr/spring2014/bbm202`) and you are responsible for the course page.**
- **The assignment must be original, INDIVIDUAL work. Duplicate or very similar assignments are both going to be punished.**
- Use understandable names for your classes, attributes and procedures.
- Obey general software engineering principles.
- Save all your work until the experiment is graded

References

[1] Live at Wembley '86. In Wikipedia. Retrieved May 04, 2014, from `http://en.wikipedia.org/wiki/Live_at_Wembley_'`

[2] The Wall – Live in Berlin. In Wikipedia. Retrieved May 04, 2014, from `http://en.wikipedia.org/wiki/The_Wall_-_Live_in_Berlin`

[3] Life Is Beautiful. In Wikipedia. Retrieved May 04, 2014, from `http://en.wikipedia.org/wiki/Life_Is_Beautiful`

[4] Input documents. Retrieved May 03, 2014, from `http://en.wikipedia.org/wiki/`

[5] Fripp, Dominic. "Using linked data to classify web documents." Aslib Proceedings. Vol. 62. No. 6. Emerald Group Publishing Limited, 2010.

[6] Cosine similarity. In Wikipedia. Retrieved May 04, 2014, from `http://en.wikipedia.org/wiki/Cosine_similarity`