ASSIGNMENT 3

Subject : Minimum Spanning Trees
TAs: Selim Yilmaz, Levent Karacan, Merve Ozdes
Due Date: 03.05.2018 23:59:59

1 Introduction

Graphs are widely used in Computer Science for a wide range of subjects. These subjects may include Natural Language Processing, Image Processing, Pattern Recognition, Speech Recognition, Bioinformatics etc.

Graphs provide a natural way of constructing relationships between objects; thereby it is a natural way of learning by using this relationship between objects. These objects may be genes in a dna, phonemes in a speech signal, pixels in an image, words in a text (see Figure 1), and so on. Depending on the type of the problem, everything can be modelled by using the graph theory. There are many operations defined on graphs. You will practice Cosine Similarity and Minimum Spanning Trees in this assignment:

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them.

A Minimum Spanning Tree is a subgraph of a given graph which has the minimum sum of the weights on the edges in the spanning tree. Minimum Spanning Trees have a wide range of application fields. For example, they are widely used in the design of the computer networks or telecommunication networks, handwriting recognition, image segmentation, clustering (see Figure 2), and so on.

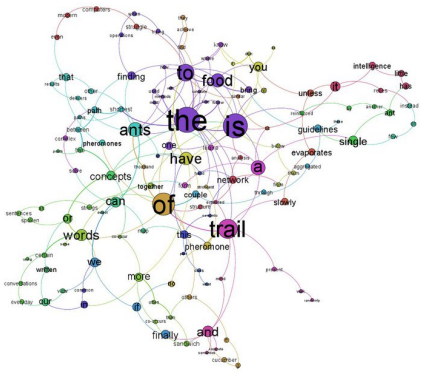

Figure 1: A word graph [3]

2 Problem Definition

In this programming assignment, you will practice on one application field of graphs. The field that you will apply graphs is Natural Language Processing. You are expected to measure semantic similarity between words by using the word vectors. Subsequently, you will find word clusters that bear semantically similar words in each of them.

Semantic similarity is a concept that measure how similar two words/documents/terms/concepts/senses are in meaning. For example, words 'blue', 'red', 'yellow' are semantically similar, whereas 'book' is no semantically similar to these words. However, book is semantically similar to 'notebook'. In Figure 3, a graph which is constructed by the Flickr tags is given (see Figure 4 for visual images for the given tags). You can see that semantically similar words are located close to each other on the graph, whereas semantically different words are quite distant from each other.

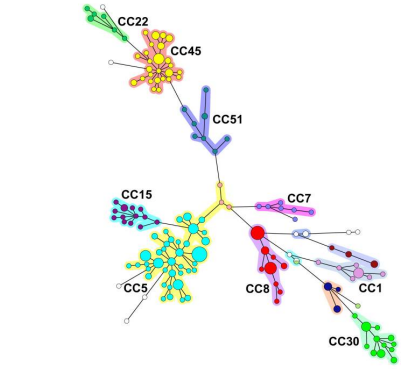

Figure 2: An example of minimum spanning tree clustering [4]

Another point that you need to observe on the graph is that, vertices in different colours refer to different clusters. Moreover, these clusters bear semantically similar words.

In this assignment, using the word vectors provided in assignment you should build a graph. Movie review text is taken as input and word vector is produced. For each word you should assign edge weights equal to the cosine similarity. You can calculate cosine similarity by using formula which is shown below.

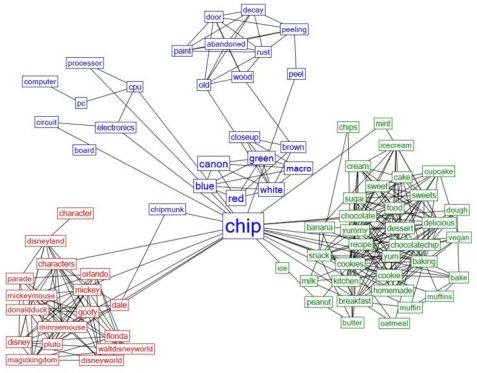

Figure 3: Disambiguation of Flickr Tags

Once the initial graph is constructed as described, you will use your graph for practicing two operations: a) measuring semantic similarity, b) finding clusters.

2.1 Measuring the Semantic Similarity Between Words

You will use your graph to measure semantic similarity between words. There are several ways to measure semantic similarity between words. However, you will use the cosine similarity:

$$cos\_sim(w_1, w_2) = \frac{\vec{w_1}.\vec{w_2}}{||w_1||.||w_2||} \quad (1)$$

In this equation w_1 and w_2 represents different word vectors. In the word vector file, each line represents a different word vector.

Figure 4: Disambiguation of Flickr Tags

2.2 Clustering

In this part of the assignment, you will use minimum spanning trees on your graph to find word clusters (see Figure 2). As indicated before, minimum spanning trees can be used for clustering data. Once the minimum spanning tree on a given graph is found,k clusters can be obtained by removing *k-1* edges that have the minimum weights on the minimum spanning tree. Therefore, k clusters are obtained. Each of these clusters are expected to have semantically similar words.

Since your initial graph is not weighted, you will construct a weighted graph before finding the minimum spanning tree. In order to create a weighted graph, you will remove all the edges from your initial graph, and you will add only the edges between the word pairs that you measured the semantic similarities in the first step. Once you have a weighted graph, you can find the minimum spanning tree and cut *k-1* edges that have the minimum weights on the graph to obtain k clusters.

The number of clusters will be provided as a parameter. Therefore, the number of clusters to be obtained will be fixed initially.

3 Input Output And Testing

You will have two input files and one output files in this assignment. All of the file names will be provided to the program as command line arguments.

- **word vector file:** A word vector is given in this file. In this word vector file, each line represents a vector of unique word in the text. That is every word and it's vector are given in different lines. A sample word vectors are given in Figure 5.
- **word pairs file:** A word pairs list is given in this file. Your program is expected to measure the semantic similarities between the word pairs. A sample word-pairs file is given in Figure 6.

```
"natural" 0.590474337339401 -0.31276223435998 -0.125205791555345 0.66697457432746
"handled" 0.637407869100571 0.11158469831571 0.0878466665744781 0.038664646446704
"criminal" -0.348169110715389 -0.175828732550144 -0.273133810609579 -0.1670994963
"medical" -0.473691284656525 -0.128665044903755 0.335133410990238 0.0202683955430
"favor" -1.57458806037903 0.138710021972656 0.959300130605698 -0.615124881267548
"too" -0.48787721991539 0.144629758782685 0.528842732310295 0.346322432160378 -0.
"king" 0.278449356555939 -0.383282765746117 0.755440235137939 -0.423472315073013
"wrote" 0.479913162067533 0.511747725307941 0.994782537221909 -0.220809370279312
"amitabh" 0.434985779225826 -0.0899343676865101 0.415430441498756 -0.450814306735
"pair" -0.299385257065296 -0.201886579394341 0.679945826530457 -0.682772636413574
```

Figure 5: A sample word vectors

```
directors-producers
film-movie
black-white
man-woman
person-man
young-woman
science-fiction
thrilling-realistic
lovely-stunning
criminals-zombies
father-son
girlfriend-boyfriend
nurse-soldier
professor-college
```

Figure 6: A sample of word pairs file

```
boyfriend,college,father,girlfriend,man,nurse,person,professor,son,woman,young
criminals,directors,fiction,film,lovely,movie,producers,science,stunning,thrilling,zombies
```

Figure 7: A sample cluster for k=2

- **cluster file:** Finally, your program will produce the minimum spanning tree and produce the clusters out of this spanning tree. 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. A sample clusters file is given in Figure 7.

4 Execution of the Program

Your implementation should run with following command:

```
java -java Assingment3.java wordVec wordpairs clusters
numberofclusters
```

Here:

- wordVec is the name of the file that has the dictionary,
- wordpairs is the name of the file that has word pairs that your program will measure how similar they are,

- clusters is the name of the file that the contents of each cluster will be written to,
- numberofclusters is the number of clusters that your program has to find by using the minimum spanning tree.

Please keep in mind that all of the names of the files will be taken as command line arguments and will not be fixed names, otherwise your program will not be able to run with other file names.

A screenshot of the terminal that shows how to run your program is given above.

You can test your program by creating your own word_pairs.

Notes

- Do not miss the deadline.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- You can ask your questions via Piazza (https://piazza.com/hacettepe.edu.tr/spring2018/bbm204) and you are supposed to be aware of everything discussed in Piazza.
- You will submit your work from https://submit.cs.hacettepe.edu.tr/index.php with the file hierarchy as below:

This file hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system)

<student id.zip>
→ src
    → *.java <FILE>
→ readme.txt (optional)

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), in whole or in part, as your own will be considered **as a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.

References

[1] Hyunjong Cho and Viet-An Nguyen. Flickr tags disambiguation, February 2011.

[2] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In in Proceedings of International Conference Research on Computational Linguistics (ROCLING X), 1997.

[3] Hoang Vu-Thien, Katia Hormigos, Gaelle Corbineau, Brigitte Fauroux, Harriet Corvol, Didier Moissenet, Gilles Vergnaud, and Christine Pourcel. Longitudinal survey of staphylococcus aureus in cystic fibrosis patients using a multiple-locus variable-number of tandem-repeats analysis method. BMC Microbiology, 10(1), 2010.