LM Data Mining and Machine Learning (2021)
Lab 1 – Text Retrieval

PART 1: TF-IDF BASED TEXT RETRIEVAL

Objective
The objective of this lab session is to apply the text-based Information Retrieval (IR) techniques which we have studied in lectures, namely:

1. Stop word removal
2. Stemming
3. Construction of the index – calculation of TF-IDF weights
4. Retrieval – calculating the similarity between a query and document

We will apply these techniques to a 'toy' corpus consisting of 112 documents – BEng final year project specifications. These project specifications were submitted by staff in Word format, but I have converted them all into plain text files for the purposes of this lab. However, I did not remove the formatting or the pieces of text which are common to all of the files.

Copy the zip archive `lab1-2021` from Canvas and 'unzip' it. You should end up with a new folder called `lab1-2021` containing all of the files that you need to complete the lab, including a folder called `docOrig` which contains 112 text files.

<u>The folder lab1-2021 will be the default folder that you work from</u>. Have a look at one of the text files in the `docOrig` folder. You should be able to identify the common formatting.

Processing of the documents
Before we can do IR we need to apply stop word removal and stemming to each of the documents in our corpus. To do this you will use two executable (.exe) files of the C programmes that are in your `lab1-2021` folder: `stop.exe` and `porter-stemmer.exe`. Note that there are also source C programmes provided in a case your computer runs on a non-Windows operating system – in that case, you will need to compile the source C programmes (`stop.c` and `porter-stemmer.c`).

Task 1: **Stop word removal**: The next task is to remove stop words from each of the documents. The 50-word stop word list `stopList50` should already be in your `lab1-2021` folder.
Now run the program `stop` on one of the documents – `AbassiM.txt` for example. To run the program, just type the below in the Command Prompt window:

```
stop stoplist50 docOrig\AbassiM.txt
```

(note that the above includes the path name to tell `stop` where `AbassiM.txt` is – this is the `docOrig` folder). This should cause a version of `AbassiM.txt` with stop words removed to be printed onto your screen. You need to store this output in a text file `AbassiM.stp`. To keep the 'stopped' documents separate from the original documents, there is created folder in `lab1-2021` called `docStop`. All of the 'stopped' documents should go in this new folder.

You need to apply `stop` to all of the project description files. To do this I have created a batch file called `stopScript.bat`, which you should have in your `lab1-2021` folder. In the Command Prompt window just type `stopScript` followed by 'return'. You need to be in the `lab1-2021` folder when you do this.

You should now have 112 files in the `docStop` sub-folder, each with a name of the form `filename.stp`.

Question 1: What is the percentage reduction in the number of words in a document as a consequence of stop-word removal? Specifically, what is the reduction in the case of the file `AgricoleW.txt`?
Answer:

Task 2: **Stemming**: The next task is to apply the porter stemmer to each '`.stp`' file. There is created another sub-folder of `lab1-2021` called `docStem`. This folder will contain a stemmed version of each file in the `docStop` folder.

Basically, for each `.stp` file you create a `.stm` file by typing, for example,

```
porter-stemmer docStop\AbassiM.stp
```

This causes a 'stemmed' version of `AbassiM.stp` to be printed on screen. You need this data to be stored in a file called `docStem/AbassiM.stm`. You need to do this for every `.stp` file. To do this I have created another batch file called `stemScript.bat`, which you should have in your `lab1-2021` folder. In Command Prompt window just type `stemScript` followed by 'return'. You need to be in the `lab1-2021` folder when you do this.

Question 2: Find the file `AgricoleW.stm`. What are the results of applying the porter-stemmer to the words `communications`, `sophisticated` and `transmissions`?
Answer:

You should now have:
- 112 original .txt documents in the folder `docOrig`
- 112 'stopped' documents in the folder `docStop`
- 112 'stemmed' documents in the folder `docStem`

Task 3: **Create the document index files:** If you've forgotten what the document index is, or what it is for, look again at the lecture slides. The next task is to create 3 index files: one for the original `.txt` documents, one for the `.stp` documents, and one for the `.stm` documents.

You should have the executable `index.exe` in your `lab1-2021` folder (or compile the program `index.c` if needed).

You should have a text file called `textFileList` in your `lab1-2021` folder. This is simply a list of all of the original .txt files – one file per line. Type:

```
index textFileList
```

followed by 'return'. After a short pause a text version of the index file will be printed on your screen. You need to store this data in a file called `textIndex`. Type:

```
index textFileList > textIndex
```

followed by 'return'. Look at this file and try to understand the information which it contains. The lecture notes will help you.

Now repeat this on the 'stopped' and 'stemmed' files:

```
index stopFileList > stopIndex
index stemFileList > stemIndex
```

Question 3: What are the document lengths (this is not the length of files in bytes – see lecture notes if you are unclear) of the documents: `docOrig\DongP.txt`, `docStop\DongP.stp` and `docStem\DongP.stm`? Why are they different?

Why is the difference between the document lengths of `docStem\DongP.stm` and `docOrig\DongP.txt` greater than the difference between the document lengths of `docStop\DongP.stp` and `docOrig\DongP.txt`?
Answer:

Question 4: The IDF of the term `adjacent` is 0.009. Why is it so close to zero?
Answer:

Question 5: Find the word `algorithm` in the three index files. Explain why the entries for this word are different in the three files.
Answer:

Task 4: **Retrieval**: The final task in this part of the lab is retrieval. To do this you will need to create a query. This is just a text file containing your query – you can create it using notepad or wordpad. An example query – `query` – should be in your `lab1-2021` folder. This query just contains the text: `communication and networks`

Next you need to apply stop word removal and stemming to the query:

```
stop stoplist50 query > query.stp
porter-stemmer query.stp > query.stm
```

You should have the executable `retrieve.exe` of the C program in your `lab1-2021` folder (or compile the source C program if needed). You can now do retrieval.

Start with the raw text files:

```
retrieve textIndex query
```

followed by 'return'. This will return a list of all the documents for which the similarity with the query is greater than 0. It also tells you the identity of the most similar document. Does the result make sense?

Now repeat this for the stopped documents and stopped query, and stemmed documents and stemmed query:

```
retrieve stopIndex query.stp
retrieve stemIndex query.stm
```

Question 6: Compare the results of these two searches with the result for the original raw text files. What do you conclude?

Task 5: Repeat Task 4 for at least 2 your own queries.

PART 2: LATENT SEMANTIC ANALYSIS

Objective
The objective of the second part of the lab is to apply Latent Semantic Analysis (LSA) to the set of BEng final year project specifications in the `docOrig` folder. Look at the notes on LSA to remind yourself about the technique, to put the following sequence of tasks into context. This part is quite challenging, so you may have problems!

Task 1: Create the Word-Document matrix
Recall that the word document matrix W is an N x V matrix, where N is the number of documents and V is the vocabulary size (the number of different words in the corpus). The nth row of W is the document vector $vec(d_n)$ for the nth document.

The executable `doc2vec.exe` of the C program will create the matrix W (or compile the source C program if needed). We will apply this program to the stemmed documents. The command is:

```
doc2vec stemFileList.txt > WDM
```

This creates a document vector for each document in the `docStem` folder and stacks them to create the matrix in the file `WDM`.

Task 2: Apply Singular Value Decomposition (SVD) to the word-document matrix
This is done in MATLAB. You will need the following commands:

```
>>W=load('WDM');
```
This reads the data in `WDM` into the MATLAB matrix `W`

```
>>[U,S,V]=svd(W);
```
This runs SVD on `W`, decomposing it as $W = USV^T$.

<u>**Question 1**</u>**:** Are the matrices `U` and `V` as you would expect? Explain.

Verify that the singular values, the diagonal elements of `S`, are ordered according to size.

<u>**Question 2**</u>: What are the values of the first 3 diagonal entries in `S`?

Now recall that the singular vectors, the 'latent semantic classes', correspond to the columns of `V`. You can access, for example, the first column of `V` and write it into the vector `sv1` by using the MATLAB command:

```
>>sv1=V(:,1);
```

Do this for the first 3 columns of `V`, creating singular vectors `sv1`, `sv2` and `sv3`.

Now you are going to try to interpret these vectors. Intuitively, the most important words that determine the interpretation of the vector sv1 are those for which the corresponding coordinate of sv1 is biggest (positive or negative).

To find the biggest positive value in sv1 we can just use:

```
>>m=max(sv1);
```

But we don't just want to know the size of the biggest number, we also need to know its position in the vector so that we know which word it corresponds to. So use:

```
>>[m,am]=max(sv1);
```

In this case m is the maximum value in sv1 and am is its index (argmax). Find the words that correspond to the three biggest values in sv1. To achieve this you need to know the order that the words occur in when the document vectors were constructed. The program doc2vec.exe is based on index.exe, and the word order is the same in both programs. So the *n*th component of a document vector corresponds to the *n*th word in the corresponding index file. Hint, the most significant word for sv1 turns out to be 'project'.

Question 3: Find the three most significant words for each of the singular vectors sv1, sv2 and sv3. What is your interpretation of the corresponding semantic classes?

END