Guillem Gili i Bueno
Enrique Reyes Illescas

Day: 20/10/2017

2nd assignment CAIM

First part

For all the tests in this part we used files from news.
First we must test the different ways to tokenize, and see which has the best behaviour. For this we picked one file from comp.os.ms-windows.misc and compared it to other files from the same folder and calculate the average. Then we did the same with another 2 different folders (40 files from each). The other files were a random selection picked only once. To run a test you must run tests.sh in terminal, and have a tests folder outside the folder. You can find the files we used in this folder.

This is in order to see which tokenization shows the highest similarity when comparing similar-themed texts, and which shows the minimal similarity when comparing different-themed texts. This may be useful to make an automatic classification when presented with two files. We don't use any filter to see which works the best by itself.

Results:
We compared comp.os.ms-windows.misc/0002413 to everything else.

Tokenization algorithm	comp.os.ms-windows.misc	comp.windows.x	rec.sport.baseball	talk.religion
Whitespace	0.0215	0.015	0.0193	0.0198
Classic	0.0223	0.0201	0.0149	0.0156
Standard	0.0224	0.0197	0.0144	0.0148
Letter	0.0282	0.0229	0.0167	0.0169

We can see that **Whitespace** fails in putting a big difference between computer related documents and baseball and even consider similar themes more different than unrelated ones. Since it falls behind in showing how different the document themes are, we won't use it for future experiments, it has proven not to work well in this situation.

Next is **Standard** and **Classic**. They both draw a big difference between computer related documents and others. We can also see that the difference between more related documents and less related documents is more noticeable when using **Standard**. This is a difficult choice, had we to choose one we would use **Standard**, since the capabilty to discern very different themes seems more important than tell small difference between similar ones.

The **Letter** filter seems to work best. After looking at the tested files, we can see that they have very few nonclassic characters, and they don't bring any new information about the word. Also since we are using English all the letters are already in unicode(no accents, minus signs) so **Standard** gets outsmarted by **Letter**.

We will keep the Letter filter as it seems the one that worked most reliably, and use it in all future experiments.

In order to test the filters, we did the same using various filter combinations: comparing a file from news/alt.atheist to 40 of the same folder and computing the average similarity, then doing the same with that file and other folders.
For **Stop** , **Lowercase** and **Asciifolding**, we did a test separately to see how much it improved the difference in similarity adding them one after another.

Filters	comp.os.ms-windows.misc	comp.windows.x	rec.sport.baseball	talk.religion
Lowercase (default)	0.0282	0.0229	0.0167	0.0169
Stop	0.0258	0.0189	0.0126	0.0134
Stop+Lowercase	0.0263	0.0219	0.014	0.0134
Asciifolding	0.0286	0.0211	0.0166	0.0178
Stop+Lowercase +Asciifolding	0.0263	0.0219	0.014	0.0134

We first tested **Stop**, **Lowercase** and **Stop** + **Lowercase**. It seems like **Lowercase** and **Stop** both work well individually, and show a clear difference between files in the same theme, files in a similar theme and non-related theme. When we combined we get a hybrid result, which ends up showing more difference between non-related themes. As we said we consider not having false positives a priority so this seems the best solution so far.

Asciifolding seems to work worse on its own as well(less similarity between related themes), and if we test it together with the others it doesn't change our solution. This makes sense as the documents we are evaluating are in English, and English has no common non-ascii symbols, this would work well in other languages, or maybe make our solution less reliable if we had scientific papers (it would ignore greek letters, commonly used in formulas and expressions).

We tried to do **Asciifolding** together with **Stop** as well to do a final check and the results we get are the exact same as in **Stop**. Thus, if we are working with English papers not containing scientific data **Asciifolding** has near null utility.

Since Asciifolding seems to have no effect in English we will only keep Stop and Lowercase filters.

To test the different stemming algorithms, we tried them separate to see which worked the best. We kept the filters and tokenization that worked best in earlier tests, to see which improved the differences the most. We used **Standard** tokenization with **Stop** and **Lowercase** filters.

Stemming	comp.os.ms-windows.misc	comp.windows.x	rec.sport.baseball	talk.religion
No stemming	0.0263	0.0219	0.014	0.0134
Snowball	0.0366	0.0266	0.017	0.0169
Porter_stem	0.0366	0.0267	0.017	0.0169
Kstem	0.0333	0.0254	0.0159	0.0155

It seems that **Snowball** and **Porter_Stem** give almost identical results, we searched in ElasticSearch documentation and we found some information regarding **Porter** being a faster equivalent of the Snowball algorithm. That would explain our results.

Finally we must compare **Porter** and **Kstem**, and we think **Porter** show a better performance. It gives a more stable average for completely non-related themes, and maximizes news on the same subject more.

So, in conclusion for this kind of files we would use a Letter tokenization, with Lowercase and Stop filters, and Porter_Stem as our stemming algorithm.

Second part:

The code was easy to understand and you could make it work without too much trouble using the theory class content. We managed to do every function on the first try by reading what it calculated, checking out the documentation and searching how to calculate it, then writing down the function.

Also in order to check if the program was working as expected, we did a test sample: we made two copies of the same novel and progressively deleted paragraphs from one of them, to see how the similarity decreased. After observing 5 cases where the result was what we expected, we concluded our code was correct.

In our scripts we use the test folder as external for commodity. We added the folder but with the tests we used but you will need to move it outside the folder in order to run the testing script (tests.sh).