Roee Ebenstein, Gaurav Soni
5243

HW1

In this homework we were required to write a code that pre-process given Reuters files, and build vectors out of them.

Regarding viewing the vectors, since the representation is complex in most cases, we do not allow viewing it, although – because it is required we dump it to files in the output directory, being visible by VI).
Instead, the vectors are public objects one can explore from a python code (in the Vector code files – you'll find a demonstration of how to do that, in the code that builds the vectors, and the vectors are the first objects that are defined in the code).

Work division:

Roee:
Reading the XML's.
Created the basic vector (with stemming and stopwords).
Created the dictionary vector.
Defined the environment (Code was written in eclipse with PyDev).

Gaurav:
Created the module for dumping files into the correct format.
Created the topic vector.
Evaluated Scipy's sparse matrices for use in the project. Currently we are sticking to Numpy's simple matrix as it's easier to write to a file and future application of the data isn't clear yet.

Approach towards SGM to XML:

The main issue is that we decided we should not extract text from the xml, but to make the SGM a valid XML.
For doing this – I first had to create a root node, Afterwards to delete non-ascii codes, and afterwards to remove non-XML characters (there are some UTF characters that are not valid within an xml).
This sweep allowed me to create a valid xml that can be loaded and read into memory, and traversed by the utilities.

These XML's are being saved in a directory for further analysis.

When reading, the XML is being queried for each "REUTERS" element child, and is being sent to the vector building algorithms.

Basic Vector

The main approach for this vector is to do the worst implementation possible, so it could be compared to other models.
Although it's the worst representation, I am still clearing non-relevant words:

Roee Ebenstein, Gaurav Soni
5243

All the words are being stemmed, and stop words are being removed.
In addition – I noticed many words in the length of 1, 2, or more than 15 characters exist, and from English knowledge – those words are mostly not relevant (garbage) – so I'm deleting them.

This is being kept in a matrix of documents words.
Since the matrix is defined by location, there are two "help" vectors that keeps track of which document and which word is in which row and column respectively.

Basic Vector with English dictionary

Since we will experiment over this in the future, I thought it would be interesting to be able to compare analysis with "junk" words (like names and gibberish) to "pure" English words, and to see if we will get the same results.

Topic Vector

This is similar to the basic words dictionary. The already cleaned XML files are leveraged to extract. Non word characters and digits are removed to create topic words. We then build a matrix with rows corresponding to files and columns corresponding to words. Two dictionaries map document to row and word to column. Further optimizations are possible in the current approach by using sparse matrices based on the application.

Sparse matrices

We are currently using basic matrices for all of our work. This is highly inefficient in memory consumption for a sparse matrix. Once the application of the data becomes clearer, we intend to use row based, column based or key based sparse matrices, from the Scipy package to get things done.