Data Mining Assignment 1

Gaurav Shah Jigar Chandra
Shah.889@osu.edu Chandra.67@osu.edu

Goal of the Assignment

To preprocess a given rudimentary data to form structured datasets and feature vectors for further processing such as automated categorization, document similarity search, and building document graphs.

Program Logic

```
Iterate through each sgm file:
    Split the sgm file into individual article
    For each article in an sgm file :
        Store Topics List, Places List
        If the article has a non-empty Body and Topic:
            Remove stop words
            Perform Stemming on each word.
            Find set of unique words for each article and Add it
                to a set: doc_list_of_set
        Else if the article has a non-empty body and Empty Topic:
            Remove stop words
            Perform Stemming on each word.
            Append null topic to a separate datastructure named
                topics
        Else if the article has an empty body and Non-empty topic:
            Append empty set to doc_list_of_set
    Reduce the word set by thresholding (Trim by 99% and 1%
        based on trial and error)
    Output word list
    Output transaction matrix
    Output binary feature vector
```

Roughly the code runs around 12-15 mins

Input Parsing and Document Frequency

Parse each Reuters article and create a dictionary containing following items for each of them:

1. Frequency dictionary: dictionary of words appeared in the article body and title as keys and the number of times they occur in the article and title as their values
2. Topics: List of topics in the article
3. Places: Places the article refers to in <PLACES> tag

We have separated articles that contain topics and those that do not contain topics as we store the articles without topics in a separate dataset, appending null to the dataset. We plan to use this dataset for further processing like missing data prediction.

Data Cleaning

We perform cleaning on all words by converting all words to lower case, and retain only the characters [a-z]. All other characters are replaced by ' '.

Stopwords filtering

Stopwords are the words that appear very frequently in the natural language and often do not have any interesting meaning from the knowledge discovery perspective. Such words should not be considered for data analysis as they can hardly be useful in determining the class of an article. We use the list of stopwords provided by Nltk to filter the words from the Reuters articles.

Stemming and Lemmatization

Stemming is a process of grouping together words that come from the same root. Using NLTK stemmer, we find out root of each word and use it instead of the original word appeared in the article.
Eg: player, playing, played, plays, play get stemmed to play
Show, showing, showed, shows get stemmed to show
The idea being applied behind the usage of these words is captured instead of the individual words which is more effective in terms of natural language analysis.
We perform this step before all other words shortlisting steps because different forms of a single stem will not be lost due to low individual frequency of stemming done in later stages.
We also tried lemmatization using wordnet and though it was as effective as stemming in reducing number of words, it is not guaranteed that it is effective in detecting root word without the context it appears in. As a result, we preferred stemming over lemmatization.

Trimming wordlist based on thresholds

Words that occur too many times and those that appear too few times do not convey much information about the documents. It is best to get rid of them and maintain those words that occur with a moderate frequency.
We experimented for various values of threshold as **{ (90%,10% : 951 words} { (95%,5%) : 219 words } { (99%,1%) : 74 words}** and identified threshold as 99% and 1% of the number of documents and we remove words that are greater than maximum threshold or less than the minimum threshold.

Feature Vectors

We created several feature vectors from the Reuters articles:

1. DOCUMENT TERM MATRIX:

Data matrix is a list of dictionaries in our case. In the representation, the first row is all the unique words across all the documents which come from the keys in each of the dictionaries and subsequent each row corresponds to one article where every value is the number of occurrences in the article. The index of the dictionary in the list of dictionaries is the article number.

Sample feature vector:

```
 ['japan', 'consider', 'miner', ‘consum', ‘dollar', ‘month', ‘four' …]
[       0,       0,              4,        0,           3,           0,          0 …]
```

2. TRANSACTION MATRIX:

Transaction matrix is a list of sets where each set provides a list of unique words appeared in each article. Whenever it is an empty set, it denotes that all the unique words which were originally present in the article have been removed by thresholding.

Sample feature vector:

DOC ID:1
TOPICS:['cocoa']
PLACES:['el-salvador', 'usa', 'uruguay']
LIST OF WORDS:set([‘weekli', ‘held', ‘publish', ‘go', ‘still', ‘rose', ‘late', ‘farmer', ‘earli', ‘good', ‘march', ‘around', ‘buyer', ‘trade', ‘name', ‘level', ‘februari', ‘januari', ‘mean', ‘certif', ‘contin’, ‘crop', ‘export', ‘expect', ‘year',])
****

3. BINARY VECTOR:

A Vector for each article, which contains an entry for each of the unique extracted words that are in the list. The entry is 1 if the word appears at least once in that article and 0 if it doesn’t.

Sample feature vector:

DOC ID:1
TOPICS:['cocoa']
PLACES:['el-salvador', 'usa', 'uruguay']
BINARY LIST OF WORDS:[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,]