# Introduction

The purpose of this lab was to analyze a folder of text files, each containing an unknown amount of XML-like news articles. The articles were to be parsed and split into feature vectors. The classes of the feature vectors were the information inside the topics and places tags within each article. Any information besides those could be used for the creation of the attributes in one or more feature vectors. I choose to rely mainly on the content in the title and body segments of the articles, as I did not see much correlation arising from the dates, and the companies, exchanges, orgs, and people tags were all empty most of the time, while title and body tags consistently held information.

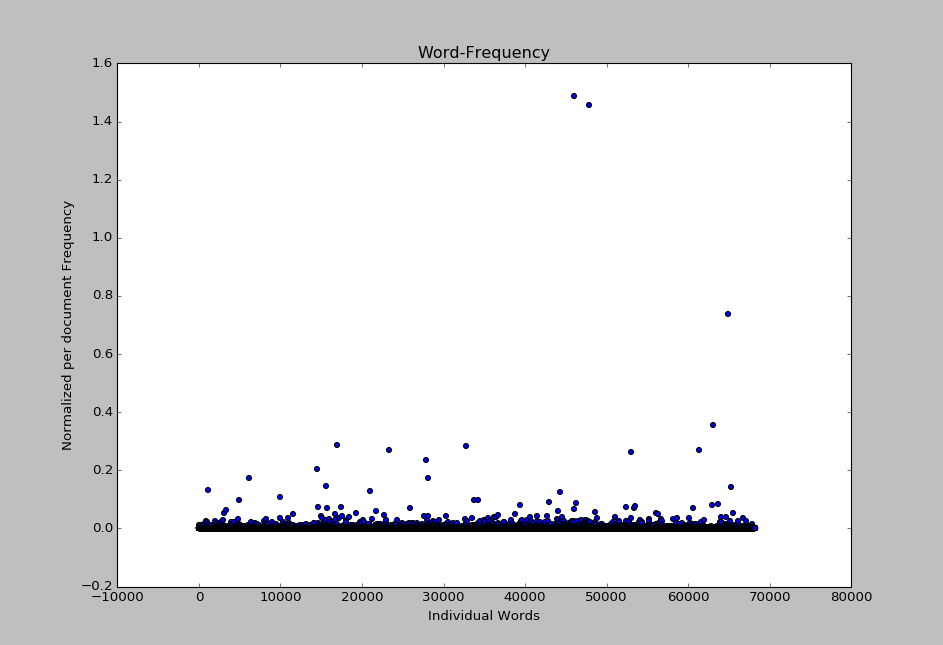
# Parsing

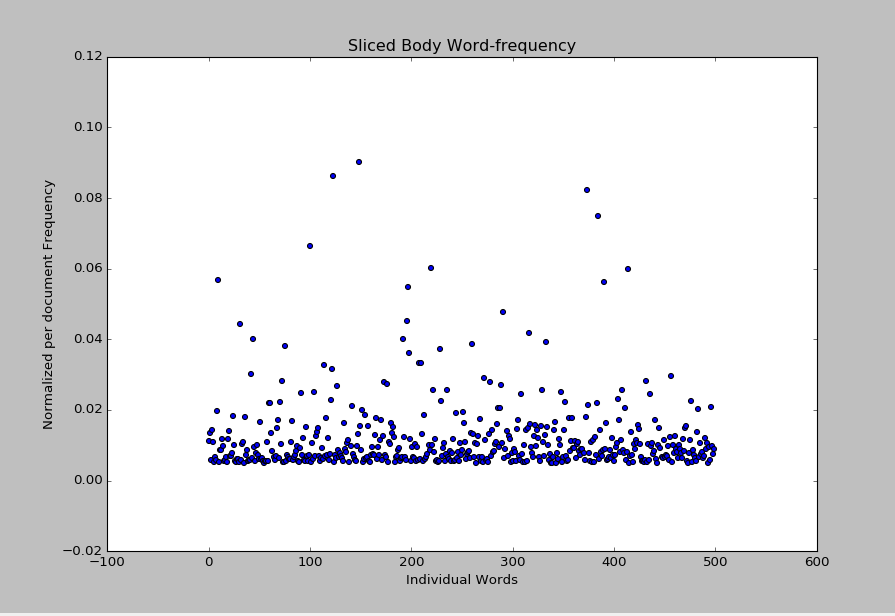
The first step to convert the text documents into modular, workable segments, was parsing the weakly formatted XML structures. I approached this in a line-by-line manner while reading the files, in order to avoid dumping entire files into memory. I did some quick searches of the documents to see where the divide was, and found the all the content was between <REUTERS></REUTERS> tags. Using the make\_router\_list\_from\_file() function, I feed in the file, and get a list of the string content between the reuters tags. From there I pass the list to the get\_entry\_array() function. This returns a list of the same length as the one passed in, now filled with tuples of the form ([topics], [places], title\_string, body\_string). Most of the parsing is done using regular expressions to find strings between tags.

# Filtering

Once the reuters articles were properly cut into topics, places, body, and title segments, it was time to begin filtering. I built three different feature vectors to use in combination to later classify articles.

**Body-text buzzword feature vector 1**

The first feature vector I decided to design was a body text buzzword vector. I pass the string of body words into the tokenize\_and\_clean() filter I designed, and it uses the NLTK module to remove stopwords and tokenize the string into a list of words (duplicates are kept for now, but this can be changed easily if it proves useful later). After tokenizing, I loop through all of the body word lists and create a dictionary of {word: total-occurrences} entries. I used this to analyze the average frequency of all of the over 100,00 unique non-stopwords in the 20,578 documents. The original scatter plot can be seen in the figure below. 

As you can see, most of the words have a very low frequency of occurrence per document, and I wanted to focus on words that were moderately rarely occurring, but not so rare that they were likely not to be seen in many documents. After seeing the number of topics (or classes) was 119, I decided to center my “buzzwords” around 1.19% frequency, so I picked a bottom threshold frequency near that at 0.05%. Then after seeing some of the topics (i.e. ‘earn’) appeared relatively often, I pushed the maximum frequency to 10% of documents. This sliced the total body words to be working with down from 101,167 to the 500 seen below..

I create an ordered list of these 500 words for the next step. Each document has its tokenized body words parsed through, and any words matching one of the 500 “buzz words” grab the index of that buzzword and append it to a new array of indices. The buzzword index array is the final representation of this feature vector. Each of these arrays vary in length from 0 to 355 terms, with the mean number of terms being 29.68.

**Body-text buzzword feature vector 2**

# The second body-text based buzzword vector I made was fairly simple. After tokenizing and cleaning out stopwords, I create a word frequency dictionary for each individual document. I then sort the dictionary and pick the top 5 words from every article and append this to the final vector dictionary. All of these feature vectors have between 1 and 5 words, 1 being if there were no body words to begin with, just the word ‘none’ is in the array.

# Topic keyword feature vector

# The second feature vector I designed was similar in design to the body-text buzzword vector, but instead of arbitrarily choosing buzzwords to be those in occurring in 0.05%-10% of documents, I choose the topic words themselves. Anytime a word in a topic appears in either the body or the title, I append it to an array. Then in much the same way as the last feature vector, I use a fixed order list of topics to reference and replace the topic buzzwords with the indices. This feature vector was often empty, but had a mean number of keywords per document over 1.