CS 221 Project Proposal
Brian Holder-Chow Lin On, Rose Perrone

http://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction

Task Definition

==========

The task is to predict the tags (a.k.a. keywords, topics, summaries), given only the question text and its title. The dataset contains content from disparate stack exchange sites, containing a mix of both technical and non-technical questions.

The Data

=======

All of the data is in 2 files: Train (7.26GB) and Test.(2.36 GB)

Train.csv contains 4 columns: Id,Title,Body,Tags

- Id - Unique identifier for each question
- Title - The question's title
- Body - The body of the question
- Tags - The tags associated with the question (all lowercase, should not contain tabs '\t' or ampersands '&')

Sample entry in train.csv:

“Id”,”Title”,”Body”,”Tags”

**"12","Crappy Random Number Generator"," This may sound like an odd question, but where can I find a random number generator that works in C or C++ that is not very good? **

** Context: I'm creating some tree graph plotting software and testing it by using multi-digit random numbers (so each digit becomes a node in the tree). The random number generator I've been using - which is the one that comes with the GNU C++ compiler - gives me a nice spread of values. That's good, but I want to see how the table looks when the numbers clump together and are less homogenous. **

** Can anyone suggest a random number generator that has been proven to be not-so-random? **

** (Oh, any anyone who links to xkcd and/or suggests I just return 4 will get sarcasm in response). **

","algorithm language-agnostic random"

Sample submission entry:

“Id”, “Tags”

6034196,"javascript c# python php java"

Test.csv contains the same columns but without the Tags, which we are to predict.

The questions are randomized and contain a mix of verbose text sites as well as sites related to math and programming. The number of questions from each site may vary, and no filtering has been performed on the questions (such as closed questions).

Literature Review
==========

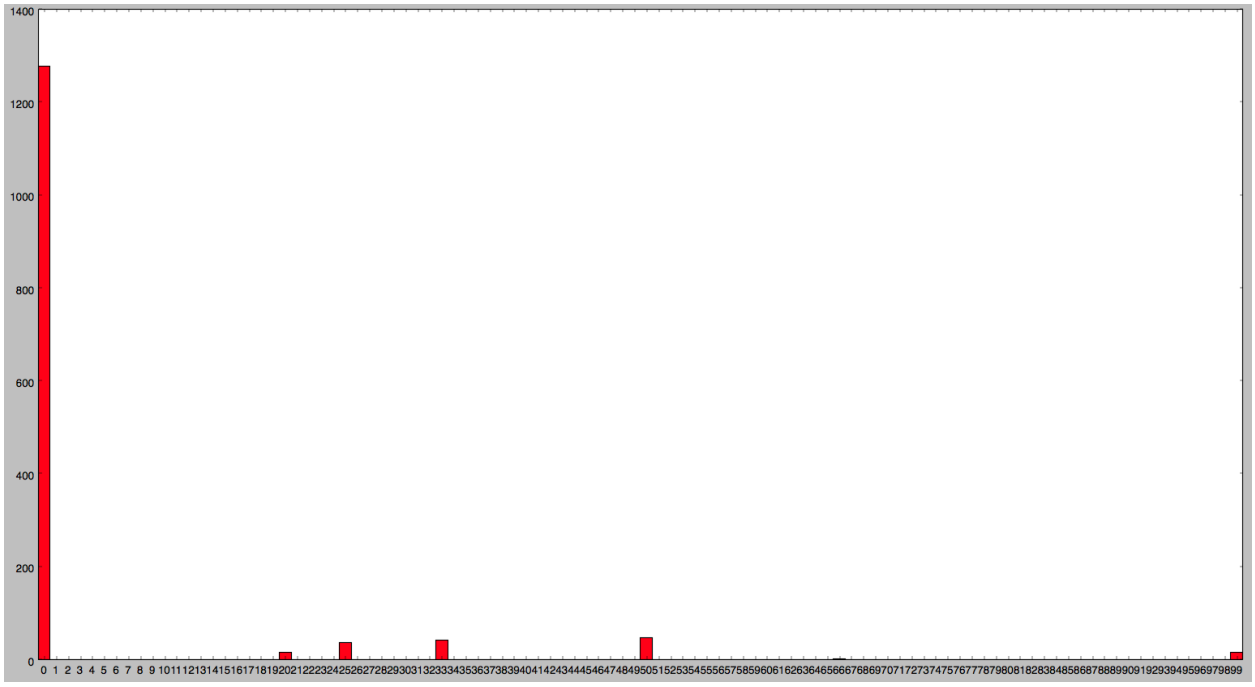
S. M. Kamruzzaman. Text Classification using Artificial Intelligence.
http://arxiv.org/pdf/1009.4964v1.pdf

Andrew Ng. Transfer learning for text classification. (2005)
http://robotics.stanford.edu/~ang/papers/nips05-transfer.pdf
This paper proposes a way to “meta-learn” a new learning algorithm that can be applied to classification problems. More specifically, their framework automates the process of finding a good parameter function for text classifiers. This framework, he says, “replaces hours of hand-tweaking with a straightforward, globally-convergent, convex optimization problem.” We could give this method a shot.

Text Classification using String Kernels (2002)
http://machinelearning.wustl.edu/mlpapers/paper_files/LodhiSSCW02.pdf

Andrew Ng, Michael Jordan. Latent Dirichlet Allocation
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Preliminary results of our baseline implementation
==========
Our baseline implementation splits the training data into about 400 files, and trains a scikit multilabel classifier on one of the files containing 2MB (about 1/4000th of the full dataset by bits). I then train the dataset on another 1/4000th of the training set. Here’s the histogram showing the classifier’s success. The y-axis is number of testing examples, and the x-axis is the percentage of tags predicted correctly.



Here's how our baseline system works:

- Load one/four-thousandth of the training data into a python array
- Strip the HTML markup and newline characters in the titles and bodies
- Merge the titles and bodies so there is one text entry to classify
- Turn the bodies of text into a matrix such that each row represents a body of text, and each column is the count of that word in the body of text. This step can incorporate bigrams, but for now, we just use unigrams.
- Use a term frequency inverse document frequency transformer that decreases the importance of frequent words like "the" and "is".
- Use a One vs Rest classifier that uses a support vector classifier.