Abstract Classification

[COMP 598 Group Project 2] *

Benedicte Leonard-Cannon
McGill University
teabennie@gmail.com

Faiz Khan
McGill University
dridon@gmail.com

Sherry Shanshan Ruan
McGill University
sherry.s.ruan@gmail.com

ABSTRACT
TBD

1. INTRODUCTION
In text classification we aim at building predictors that classify text documents into a specific category from a given set. Typically this is done by taking an input of text documents and labels for the categories and using those train classifiers to be able to predict those text documents. Our use case specifically deals with classifying an input set of approximately 96,000 abstracts that are labelled as either being from cs, stat, math or physics fields.

In order to use standard classifiers for these situations we first need to transform the text in to inputs that may be used with standard machine learning algorithms. Many feature engineering strategies such as bag-of-words, bi-grams and n-grams are available. We employed the bag-of-words strategy in the construction of our feature set.

In preprocessing we tokenize our data using two lexers. We then run a series of filters and transformations. We calculate the number of occurrence of all the words and drop all of the words that occur less than five times in order to cater to limited computational resources. We select our features using univariate feature selection to get the 2,000 words with highest variance using chi-squared. We do this twice, once using stemming and once without. We then generate four datasets with the 2,000 words, binary with/without stemming and bag-of-words with/without stemming. The binary set simply sets a 1 if the word occurs otherwise it leave it at 0 and the bag-of-word sets show the number of occurrences in the abstracts.

— Quick mention on classifiers and cross validation

— Potential uses What is text classification (or categorization) Briefly describe our methodology Talk about potential uses?

2. RELATED WORK
In the following section, we present existing work conducted by other researchers in the task of text categorization.

Genkin and Lewis [1] proposed a Bayesian lasso logistic regression model for binary text categorization that relied on a Laplace prior to reduce the risk of overfitting. Their approach addressed the impracticality of fitting a standard logistic regression model to a dataset containing a large feature space. More precisely, their training algorithm used prior probability distributions of the model parameters to encourage model sparsity. According to them, this approach produced a compact model that is effective and does not overfit. In practice, their algorithm performed as well as two state-of-the-art categorization models (support vector machines (SVM) and ridge logistic regression) on five standard test sets (ModApte, RCV1-v2, OHSUMED, WebKB and 20 NG).

Joachims [2] was the first to study the performance of SVMs for text classification in his 1998 paper. Joachims used two SVMs-one based on a polynomial kernel and the other on a radial basis function (RBF) kernel-. He compared both of these models with the following benchmark algorithms: Naive Bayes, Rocchio, k nearest neighbors (k-NN) and C4.5 decision tree. Here again, the performance of these classifiers were assessed through the ModApte and Ohsumed datasets. Prior to fitting, these datasets were reduced to a bag-of-words representation out of which stop-words were discarded. The resulting feature vectors were normalized to unit length and the best features were selected according to their information gain. From the experiments conducted, Joachim concluded that both SVM algorithms outperformed the four benchmark algorithms significantly.

3. METHODOLOGY
There is a significant amount of information contained within the collection of abstracts. In order to reduce these into inputs that could be used for standard classifiers, many feature engineering strategies such as bag-of-words, bi-grams and n-grams are available. We employed the bag-of-words strategy in the construction of our feature set.

Bag of Words The bag of words strategy takes a collection of documents and ideally outputs a dictionary of words with a subset of words (not necessarily a proper subset) as keys and a set of ranks from 0...k-1 for a bag of k words. If there are d documents and k words in our samples we create a matrix of shape k x d. Each document is then scanned for each of the k words in the bag and the number of occurrence of the word (or simply that it occurs) are recorded in one row as a sample. The matrix is used to then train our classifiers.

*The complete dataset of this report is available at `http://www.acm.org/eaddress.htm`

The implementation of the algorithm described in this report is avaiable at `http://www.acm.org/eaddress.htm`

In order to construct the bag-of-words, we had to go through a series of preprocessing steps and a final feature selection step. We now discuss these below:

Data preprocessing The abstracts were presented in LateX code. This presented an extra challenge on top of typical text processing. We first tokenized the data using a couple of lexers. We then removed punctuation and stop words. We then generate a dictionary of all words encountered throughout the abstracts as the keys with their total occurrence. We do this dictionary with stemming and without stemming. We present this series of steps in more detail:

Latex Parsing: Library: Pygments We used the latex lexer presented in the pygments [reference]. We modified it to detect alphanumeric words (simply referred to as words for this subsection) in text and commands. We implemented a new lexer using the pygments api to tokenize plain text. We took the tokens from the latex parser and for those identified as words we ran them through the plain text lexer. The plain text lexer identified stopwords, punctuation, numbers and words.

Filtering: Stopwords are common words that occur the context of the documents being processed. We took the most common english words such as âĂIJtheâĂİ or âĂIJifâĂİ and removed them from our input. We also removed all punctuation, latex equations and commands from the inputs as well. Lastly, we filtered out words with less than three characters.

Transformation: Library: nltk (Natural Language Toolkit) After filtering, we transformed all words to their lowercase equivalents. We then output two sets after this, one with stemming and without stemming. Stemming is the reduction of words such as âĂIJproducingâĂİ and âĂIJproduceâĂİ to âĂIJproducâĂİ. Stemming puts words that are similar together reducing the size of features. However, depending on the aggression of stemming, its possible to loose information or corrupt some pieces of data for example âĂIJpunctualâĂİ and âĂIJpunctuationâĂİ might yield âĂIJpunctâĂİ by a very aggressive stemmer. We used the nltk snowball stemmer for our stemmed dataset [reference].

Word Count Dictionary: At this point we had over 100,000 unique words for non-stemmed and over 70,000 unique words for stemmed. Given that there are 96,000 samples in our input, we could not generate the counts for all possible words and run it through feature selection due to memory limitations. We created a dictionary of all 100,000 words and their total occurrence throughout the input set. We dropped all possible words that occurred < 5 times throughout all samples. From here we had around 30,000 words to deal with and we proceeded to feature selection.

Feature Selection: We first used all 30,000 words and generated the number of times they occurred in each abstract to create a matrix of samples against features. Each abstract had one row where the column showed how many times given word for that column occured in that specific abstract. Since we have over 30,000 words and number of words abstracts are in the few hundreds at most then its easy to see that this is a very sparse matrix. We ran a univariate selection algorithm using the chi-squared distribution to get a set of 2,000 words with the highest variance. We did this for both stemmed data sets and unstemmed datasets. We used these 2,000 words in order of presentation by chi-squared as bag-of-words.

Feature Generation: With our bag of 2,000 words we now go through all the abstracts, preprocess them again and then create a matrix with the 2,000 words as features on the column and their occurrences in the abstracts as the rows. Again we do this for both stemming and non stemming cases. We also generate this data again but instead of number of occurrences of each of the 20,000 words in the abstracts we simply mark a boolean to indicate if it occurred or not. Our final dataset thus has six files. Two bag-of-word files, with words appearing in order of importance, one with stemming and one without stemming. And four feature sets generated from using each of the bag-of-words: binary with stemming, binary non-stemmed, bag-of-words stemmed, bag-of-words non-stemmed.

4. TESTING AND VALIDATION

detailed analysis of results, NOT Kaggle

5. DISCUSSION

Improvements -Combine bag-of-words with bigrams or trigrams -Normalize the feature vectors by abstract length (here not a big difference since all abstracts are roughly the same length) -Consider formulae (might be easy to map a given formula to a particular field!)

We hereby state that all the work presented in this report is that of the authors.

6. REFERENCES

[1] Genkin, Alexander, Lewis, D. David, Madigan, and David. Large-Scale Bayesian Logistic Regression for Text Categorization. *Technometrics*, 49(3):291–304, Aug. 2007.

[2] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nedellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Heidelberg et al., 1998. Springer.