# Text Classifier for dbGaP

July 11, 2012

**Abstract**

In this work, we build a text-classifier for retrieving relevant documents from dbGaP. We empirically evaluate two different classificaiton methods, and achieve much better accuracy than traditional keyword based search approaches.

## 1 Introduction

In this work, we address the issue of finding relevant information from *dbGaP*. We demonstrate the shortcomings of keyword based search, and develop a text classifier that overcomes these shortcomings. Section 2 describes our dataset. Section 3 describes the models we use for classification. Section 4 describes the design of our experiments to learn the parameters of the classifier. Finally, we present experimental results in section 5 and discuss them qualitatively in section 6.

## 2 Dataset

### 2.1 Selection of Reports

The database of Genotypes and Phenotypes (dbGaP) developed by National Heart, Lung, and Blood Institute (NHLBI) hosts genome wide association studies (GWAS) data. The dataset comprises 264 studies as of April, 2012. We collected the abstracts of these studies to form our dataset. We refer to an individual abstract in this dataset as a *document*.

### 2.2 Document Labeling

We assign upto 7 different *labels* to each document. These labels are shown in table 1, and were assigned manually depending on its relevance to the document. For e.g. we assign the label *heart* to all documents that are relevant to heart-related ailments.

Note that many documents are assigned multiple labels because they may be relevant in more than one specific field of interest. For e.g. all documents assigned the label *lungcancer* are also assigned the label *lung*, but not vice versa.

### 2.3 Document Representation and Preprocessing

**Document Representation.** We use the *bag-of-words* representation of documents. After discarding words that occur in only one document, and words that occur just once in the dataset, the vocabulary size reduces to 3078.

| Label Number | Label Name | Number of Documents Assigned |
|---|---|---|
| 1 | Heart | 47 |
| 2 | Atherosclerosis | 22 |
| 3 | Blood | 19 |
| 4 | Lung | 17 |
| 5 | Diabetes | 15 |
| 6 | Other | 88 |
| 7 | General | 16 |

Table 1: Labels assigned to documents in the dataset

**Preprocessing.** Each abstract is pre-processed to remove numbers, punctuation marks and non-ascii characters. In the bag-of-words representation, we maintain normalized word counts for each of the 3078 words, scaled in the range $[-1, +1]$. Each document $x_i$ also has 8 corresponding labels $\{y_{ij}\}_{j=1}^{7}$ where $j$ corresponds to the label number in table 1.

A label $y_{ij}$ takes the value $+1$ if label number $j$ is assigned to the document $x_i$, and takes the value $-1$ otherwise.

# 3 Methods

Support Vector Machines (SVM), when applied to text classification, have been shown to combine high performance and efficiency with theoretical understanding and improved robustness [1]. In particular, it is highly effective without greedy heuristic components. In this work, we evaluate a linear soft-margin SVM classifier and a traditional Naive Bayes Approach to text classification.

Consider a dataset $\{\vec{x}_i, \vec{y}_i\}_{i=1}^{n}$ with $n$ examples, where $\vec{x}_i$ is a document and $\{y_{ij}\}_{j=1}^{k}$ its label such that $y_{ij} \in \{-1, +1\}$, and $\vec{x}_i \in \mathbb{R}^d$. Here, $d$ is the size of the vocabulary that represents document $x_i$.

In order to predict *one* binary label $y_{ij}$, we consider a classifier function $f(x_i) = g(x_i; w)$ where $w$ is the parameter to be trained. We assign a label $\hat{y_{ij}} = +1$ iff $f(\vec{x}_i) > t$, and $\hat{y}_{ij} = -1$ otherwise, where $t$ is a configurable threshold.

## 3.1 Linear soft-margin SVMs

In a linear soft-margin SVM, $g$ is the dot product function. The objective of learning is to find

$$w = \arg\min_{w \in \mathbb{R}^d} \| w \|^2 + C \sum_{i=1}^{n} loss(f(x_i), y_i) \tag{1}$$

where *loss* refers to the loss function. Also, $C = \frac{1}{n\lambda}$, where $\lambda$ is the strength of regularization. We use *hinge-loss* as a candidate loss function for our experiments. Hinge loss is given by

$$loss(f(x_i); y_i) = \max\{0, 1 - y_i f(x_i)\}$$

In (1), $w$ is a vector of parameters that defines the space of candidate functions. Both $x_i$ and $w$ have the same dimensionality $d$. Note that a small value of $C$ corresponds to strong regularization.

### 3.2 Naive Bayes

It has been shown that multinomial naive-bayes for text classification performs better than bernoulli naive bayes when the size of the vocabulary is greater than 1000 [2].

In this model, we estimate the probability

$$P(y_{ij} = 1 \mid \vec{x}_i) \propto P(y_{ij}) \prod_{k=1}^{d} P(x_{ik} \mid y_{ij})$$

Where the constant of proportionality is $1/P(x_{i1},x_{i2},...,x_{id})$. The classification rule then becomes:

$$\hat{y}_{ij} = \arg\max_{z} \left[ P(z) \prod_{k=1}^{d} P(x_{ik} \mid z) \right]$$

## 4 Design of Experiments

### 4.1 Train-test split and Cross Validation

We divide the dataset into two parts - a training set, comprising 67% of the examples, and a test set comprising the remaining 33%. This division is done in a stratified way to ensure that the distribution of output label in the test set is approximately the same as in the training set.

Within the training set, we perform 5-fold cross validation to tune the design parameters and prevent overfitting. In doing so, we again perform stratified splits of the training set, where each fold is selected such that the distribution of the output label is approximately same in all the folds. This allows us to train unbiased design parameters that will produce a better model.

### 4.2 Regularization Strength

The regularization strength parameter $C$ is selected from the set of candidate values $S = \left\{2^{-14}, 2^{-13}, ..., 2^{13}, 2^{14}\right\}$ ,where the optimum value is selected via cross-validation.

The technique used to select the optimum value of $C$ is *Grid Search*, where we search through the set of all candidate values in $S$. For each $C' \in S$, we train a model on 90% of the training set and obtain the loss on the validation set. We do this exhaustively for all the 10 folds in the training set to obtain the average loss for $C'$. The optimum value $C^*$ is the one with the lowest loss. Note that for SVM methods, the loss function we use is hinge loss, while for Naive Bayes, we use negative log-likelihood.

## 5 Experimental Results

Each prediction of the classifer can be either a true positive ($tp$), true negative ($tn$), false positive ($fp$) or false negative ($fn$). We evaluate the performance of our classifier using the following metrics:

- $Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$
- $Precision = \frac{tp}{tp+fp}$
- $Recall = \frac{tp}{tp+fn}$

- $F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

The comparison of different classification methods on the different labels in our dataset are shown in table 2. The results obtained via manual keyword-based searches are shown in Table 3. The results show that text classification can produce models that give much better performance, especially in terms of precision.

Figure 1 shows the relative F-measure scores of the two text classifiers, and figure 2 shows the performance of each individual metric - Accuracy, Precision and Recall. These graphs show that a linear soft-margin SVM classifier outperforms a Naive Bayes classifier in all of these metrics.

| Label | Linear Soft-Margin SVM | | | | Naive Bayes | | | |
|---|---|---|---|---|---|---|---|---|
| | Accuracy | Precision | Recall | F-Measure | Accuracy | Precision | Recall | F-Measure |
| Heart | 0.9318 | 0.8462 | 0.7333 | 0.7857 | 0.8523 | 0.7500 | 0.2000 | 0.3158 |
| Atherosclerosis | 0.9659 | 0.8333 | 0.7143 | 0.7692 | 0.9545 | 0.8000 | 0.5714 | 0.6667 |
| Blood | 0.9431 | 0.6667 | 0.5000 | 0.3333 | 0.9091 | 0.2500 | 0.1667 | 0.2000 |
| Lung | 0.9659 | 1.0000 | 0.4000 | 0.5714 | 0.9204 | 0.3333 | 0.1667 | 0.2222 |
| Diabetes | 0.9545 | 0.6667 | 0.4000 | 0.5000 | 0.8977 | 0.1667 | 0.2000 | 0.1818 |

Table 2: Comparison of the Results using different classification methods

| Label | Keyword Based Search (Manual) | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F-Measure |
| Heart | 0.6386 | 0.3071 | 0.7222 | 0.4309 |
| Atherosclerosis | 0.9333 | 0.5714 | 0.8333 | 0.6780 |
| Blood | 0.4070 | 0.0730 | 0.7647 | 0.1333 |
| Lung | 0.6561 | 0.1388 | 0.7500 | 0.2344 |
| Diabetes | 0.6526 | 0.1250 | 0.9333 | 0.2205 |

Table 3: Performance of Keyword based searches (done manually).

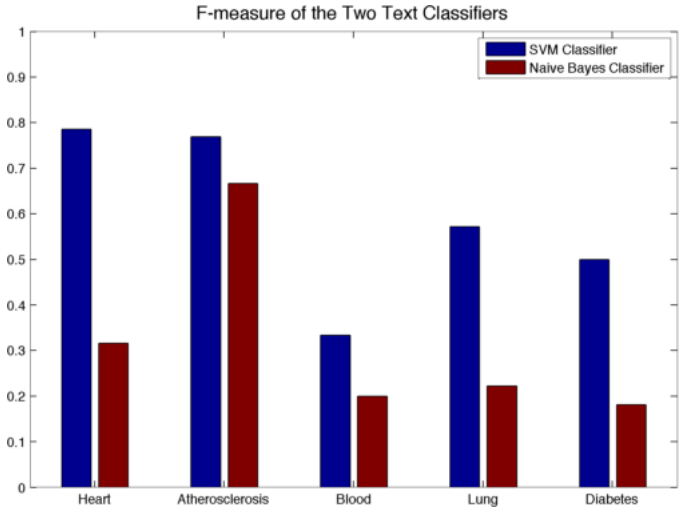


Figure 1: Relative Performance of the Two Text Classifiers measured by F-measure.

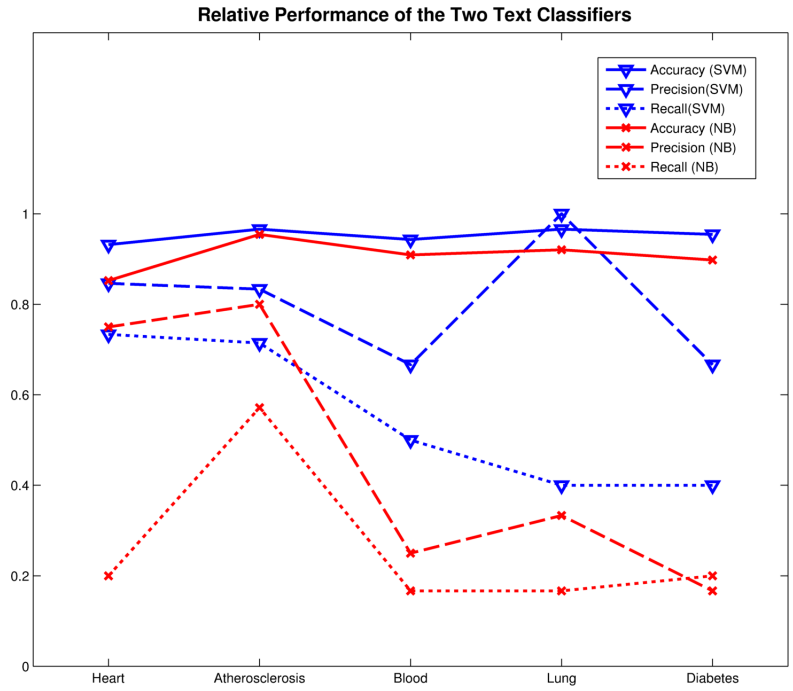


Figure 2: Relative Performance of the Two Text Classifiers in terms of Accuracy, Precision and Recall.

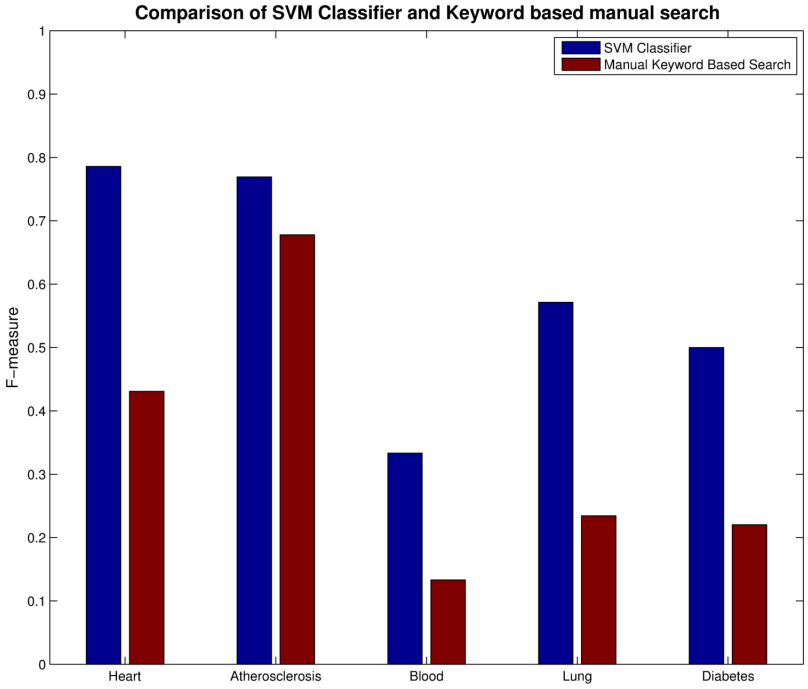


Figure 3: Comparison of SVM Classifier and Keyword based manual search.

# 6 Discussion (Incomplete)

**Points:**

- We have shown that an SVM based classifier is much better than manual keyword based classifier on all metrics.
- This is a preliminary work, and we need to collect more data and build a better text-classifier
- Limitations:
  - Didn't evaluate other models
  - No stemming
  - Didn't eliminate stop-words
  - More sophicated feature extraction is not done (e.g. bigrams, UMLS Concepts, POS Tagging, etc.)

# References

[1] Thorsten Joachims. 2002. Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms. Kluwer Academic Publishers, Norwell, MA, USA.

[2] Andrew McCallum and Kamal Nigam. 1998. A comparison of event models for Naive Bayes text classification. In Proc. AAAI-98 Workshop on Learning for Text Categorization, pages 41–48. AAAI Press.