CS224N Final Project
Probabilistic Smoothing Models for Automatic Text Classification

Sep Kamvar and Carla Pinon

June 6, 2001
Contents

1 Introduction .................................................. 1
  1.1 Practical Motivation ........................................ 1
  1.2 Theoretical Motivation .................................... 1
  1.3 Outline of Task ............................................ 1

2 Task .......................................................... 2
  2.1 Maximum Likelihood Estimate ............................. 2
  2.2 Bayesian Estimation with Uniform Prior ................. 3
  2.3 Latent Semantic Indexing .................................. 3
  2.4 Probabilistic Latent Semantic Indexing ................ 3
  2.5 Second-Order Context Representation ................... 4

3 Experimental Methods ...................................... 4
  3.1 Data ...................................................... 4
  3.2 Standard Classification ................................... 5
  3.3 Cross-Validation to determine Optimal Number of Factors ..... 5
  3.4 Performance Evaluation: Precision-Recall and F-measures ...... 6

4 Results ....................................................... 7

5 Conclusions .................................................. 12
1 Introduction

1.1 Practical Motivation

We seek to address the task of automatic classification of medical abstracts. Currently, the National Library of Medicine employs human taggers to read and hand-classify medical abstracts in the Medline digital database. This costs the NLM an estimated 1 million dollars per year. While some research is currently being done on automatically assigning MeSH terms to medical abstracts, much of the work in this area uses crude lexical or statistical methods, and we believe that using more sophisticated methods can substantially improve performance.

1.2 Theoretical Motivation

The fundamental problem of statistical learning theory is that of generalization performance (i.e. the error rate on test sets). For a given learning task, with a finite amount of training data, the best generalization performance will be achieved if the right balance between accuracy and capacity is achieved (where accuracy is defined as the error rate on the training set, and capacity is defined as the ability to learn any training set without error). The problem of too much capacity is often referred to as overfitting, and so the goal of machine learning can be said to be to achieve high accuracy on training sets without overfitting. Standard Naive Bayes classifiers suffer from the problem of overfitting, which is especially detrimental to performance in light of the sparsity of data in Natural Language applications. This has led to the introduction of various smoothing methods for use with Naive Bayes classifiers. We argue that the poor performance of Naive Bayes classifiers in comparative evaluations is due to the problem of overfitting and the crude nature of the probabilistic smoothing methods used with Naive Bayes classifiers, rather than the ”naive” independence assumptions of these classifiers. We show that, when used with sophisticated and theoretically sound smoothing techniques, Naive Bayes classifiers can perform competitively with some of the best-performing pattern classification algorithms well (such as memory-based learning) for document classification tasks. We use these methods to assign MeSH terms to medical abstracts, and evaluate their performance in comparison other methods for automatically assigning MeSH-terms.

1.3 Outline of Task

We extend our work on the word sense disambiguation project by developing, implementing, and comparing several different smoothing methods for classification tasks. We
compare a variety of different smoothing methods, including various Bayesian Estimators (such as Laplace smoothing), generative models based on reduced-rank matrix approximations (such as truncated Singular Value Decomposition and Probabilistic Latent Semantic Indexing), and more linguistically motivated smoothing methods (such as second-order context representation). We compare the performance of Naive Bayes Classifiers using these smoothed training probability distributions to variants of instance-based learning for both word-sense disambiguation and MeSH term indexing.

2 Task

Text classification is one of the more difficult machine learning problems, since it deals with very high-dimensional data sets with arbitrary patterns of missing data. Because of this inherent sparseness of data, choosing good statistical estimators for words is extremely important to the performance of a text classification system. This problem has also been presented and treated under the guise of term-weighting, feature selection, and probabilistic smoothing. Here, we explore five different statistical estimators, and evaluate them according to the classification performance using Naive Bayes. We explore the following five statistical estimators for word frequencies:

- Maximum Likelihood Estimate
- Bayesian Estimation with a uniform prior (Laplace’s law)
- Truncated SVD (Latent Semantic Indexing)
- Aspect Model (Probabilistic Latent Semantic Indexing)
- Second-Order Context Representation

2.1 Maximum Likelihood Estimate

The maximum likelihood estimate uses the relative frequency as a probability estimate for each word. This is the estimate which gives the highest probability to the training corpus. The maximum likelihood estimate is given by:

\[ P_{\text{MLE}}(w) = \frac{C(w)}{N} \]  

(1)

MLE makes the probability of observed events as high as possible subject to standard stochastic constraints. While this will work well where the data is dense, it should not work so well for highly sparse data.
2.2 Bayesian Estimation with Uniform Prior

A Bayesian Estimation with Uniform Priors, also known as Laplace’s Law, addresses the sparse data problem by assuming a uniform or uninformative events. It is given by:

\[ P_{Lap}(w) = C(w) + 1/(N + B) \]  

(2)

2.3 Latent Semantic Indexing

The Singular Value Decomposition is perhaps the most informative matrix decomposition in linear algebra. The factorization is given by

\[ A = U \Sigma V^T \]  

(3)

where U and V are constrained to be orthogonal matrices, and \( \Sigma \) is constrained to be a diagonal matrix. For every matrix A, there exists a unique SVD. The diagonal elements of \( \Sigma \) are called the singular values, and they are generally ordered in decreasing numerical order. One property of the SVD is that the truncated SVD (i.e. if we set the smallest n-k singular values=0) is the best rank k approximation to the original matrix in the L2 or Frobenius norm. For this reason, reduced-dimension SVD has been used for eliminating noise in data, and for smoothing in general. We smooth our original term-document matrix by taking the reduced-dimension SVD and projecting the original document vectors onto the reduced dimension subspace defined by the singular vectors U. The LSI estimator is given by

\[ P_{LSI}(w) = \frac{\sum_d A(A^T A)^{-1} A^T d_w}{\sum_d \sum_w A(A^T A)^{-1} A^T d_w} \]  

(4)

where d is the document vector corresponding to document d, and A is given by:

\[ A = U \hat{\Sigma} V^T, \hat{\Sigma} = truncated(\Sigma) \]  

(5)

2.4 Probabilistic Latent Semantic Indexing

Probabilistic Latent Semantic Indexing (PLSI) is an aspect model where the co-occurrence probability \( P(w, d) \) (the probability of word w occurring in document d) is defined by:

\[ P(w, d) = \sum_z P(z)P(w|z)P(d|z) \]  

(6)

Like the truncated SVD, PLSI is also a dimension reduction method. Each latent class z can be seen as a factor which defines a probability distribution over terms and documents.
Each document can be described by its distribution over component factors (rather than its distribution over its component terms. The PLSI estimator $P(w)$ is given by:

$$P_{PLSI}(w) = \sum_d \sum_z P(z)P(w|z)P(d|z)$$

(7)

2.5 Second-Order Context Representation

In SOCR, we represent each word as the vector of the words that co-occur with it. We represent each document as the centroid of its component word vectors. The SOCR estimator is given by:

$$C_{SOCR}(w) = (\sum_d \frac{1}{n_d} \sum_{w' \in d'} (AA^T)'_w)w$$

(8)

$$P_{SOCR}(w) = \frac{C(w)}{\sum_w C(w)}$$

(9)

3 Experimental Methods

3.1 Data

We begin by downloading two collections of Medline abstracts from the National Library of Medicine’s PubMed. The first collection, abbreviated as “scz”, refers to a collection of Medline abstracts relating to schizophrenia. These abstracts were gathered by doing a search through PubMed of the following MeSH terms – “schizophrenia” AND “Adult” AND “Antipsychotic Agents” AND “drug therapy” AND “Human”. We do this in order to ensure that all abstracts collected will have at least all five MeSH terms in common. This guarantees that we do not get an impoverished data set, i.e., during the testing phase, we will not come upon an abstract with MeSH terms we have never encountered before during the training phase. Similarly, we gather a second collection, which we call “p53”, referring to abstracts relating to chromosome p53 and cancer. These abstracts were gathered by doing a search on the keyword “p53” AND the MeSH terms “female” AND “breast neoplams” AND “genetics” AND “Human” and “genes, p53”. We limit our document collection to 350 documents each. We split each collection into two parts – the training set (80%, or 280 abstracts) and the testing set (20%, or 70 abstracts) and generate an answers set based on the testing set. The answers set contains the title of each abstract followed by the list of MeSH terms (one MeSH per line) that were manually assigned to it by the categorizers at PubMed. It must be noted that in Medline, MeSH terms appear under two tags – “Descriptor” and “SubHeading” and that a given MeSH term can appear under both tags any number of times, thereby giving rise to duplicate MeSH terms. For the purpose of this study, we treated each tag equally
and simply created a generic tag called “MeshTerm” that encapsulates both tags. We retain duplicate MeSH terms as they do not affect the training/testing phase.

3.2 Standard Classification

Depending on the classification method used, our approach to the automatic assignment of MeSH terms to a Medline abstract consists of either one or two processes – the MeSH term classification process and the parameter selection process. The four classification methods of Naive Bayes (NB) with no smoothing, Naive Bayes with LaPlace (or add-one) smoothing, Naive Bayes with second-order context representation (SOCR) smoothing and K Nearest Neighbors (KNN) are straightforward and merely consist of the MeSH term classification process. That is, we have each of the four methods learn the criteria for MeSH term assignment by feeding the scz and p53 training set as input. After this training phase, we feed the scz and p53 testing set as input to each of the four methods and compare the automatically assigned MeSH terms to the correct MeSH terms (i.e., the MeSH terms in the answers set which are the MeSH terms that were manually assigned).

3.3 Cross-Validation to Determine Optimal Number of Factors

The two classification methods of Naïve Bayes with Singular Value Decomposition (SVD) and Probabilistic Latent Semantic Indexing (PLSI) smoothing are not as straightforward. Both are generative models based on reduced-rank matrix approximations and thus require an operational parameter C (the number of classes) that determines the new (reduced) rank of the input matrix. This parameter C must be selected in advance before the main MeSH term classification process. The parameter selection task is a simple optimization problem. Specifically, the value for C is chosen individually for SVD and PLSI smoothing given the data set so that the performance by both classification methods is optimized. The parameter selection process is carried out only once off-line at the beginning. We do this by further splitting the training set into two parts – the tuning set (80%, or 224 abstracts) and the tune testing set (20%, or 56 abstracts). A tune answers set is likewise generated based on the tune testing set. The parameter selection process operates according to the MeSH term classification process, that is, after choosing a set of parameter values we want to test on, we use the tuning set as a training set and then evaluate the performance of each of the two classification methods on the tune testing set. The parameter value that performs the “best” is chosen as the optimal parameter value for that particular classification method and data set collection and is then used during the actual MeSH term classification process, i.e., the “real” training/testing phase. For the purposes of this study, we tested on the following eleven values for parameter C: 10, 25, 50, 75, 100, 125, 150, 175, 200, 215, 224. We cap the
value of C at 224 since our tuning set contains at most 224 abstracts.

3.4 Performance Evaluation: Precision-Recall and F-measures

The MeSH terms that are automatically assigned by each of the six classification methods are outputted in ranked order, that is, if an abstract was assigned m MeSH terms, then the first MeSH term that appears is the most likely MeSH term (has the highest probability score) followed by the next and so on until the mth (last) MeSH term which is the least likely MeSH term (has the lowest probability score). Evaluation of the performance of the six classification methods is done by comparing the automatically assigned MeSH terms to the correct (manually assigned) MeSH terms and calculating two common measures, namely recall and precision, which are defined as follows:

\[
\text{recall} = \frac{tp}{tp + fn} \tag{10}
\]

\[
\text{precision} = \frac{tp}{tp + fp} \tag{11}
\]

We define:

- \( n \) = number of manually assigned MeSH terms
- \( m \) = number of automatically assigned MeSH terms
- \( R = r_1, r_2, r_3, ..., r_n \) = set of manually assigned MeSH terms
- \( P = p_1, p_2, p_3, ..., p_m \) = set of automatically assigned MeSH terms
- \( tp \) = true positives (the number of MeSH terms assigned to the abstract both manually and automatically) = count of \( p_i \) in \( R \) = count of \( r_i \) in \( P \)
- \( fp \) = false positives (the number of MeSH terms assigned to the abstract automatically but not manually) = count of \( p_i \) not in \( R \)
- \( fn \) = false negatives (the number of MeSH terms assigned to the abstract manually but not automatically) = count of \( r_i \) not in \( P \)

More specifically, for each abstract in the testing set:

1. Initialize \( R \) to contain all the manually assigned MeSH terms of the abstract, i.e., \( R = \{r_1, r_2, r_3, ..., r_n\} \)
2. Initialize \( P \) to be empty
3. Take the first automatically assigned MeSH term and add it to $P$, i.e., $P = p_1$

4. Calculate recall and precision given $R$ and $P$

5. Take the next automatically assigned MeSH term and add it to $P$

6. Calculate recall and precision given $R$ and new $P$

7. Repeat steps (5) and (6) until $P$ contains all automatically MeSH terms, i.e., $P = p_1, p_2, p_3, ..., p_m$

At this point, we have $m$ recall/precision pairs for each abstract. Duplicate recall values are handled by taking the first recall/precision pair and removing any subsequent recall/precision pairs that have the same recall value. This gives us the highest precision value for any given recall value since we are adding automatically assigned MeSH terms to our set $P$ in decreasing rank. Removal of duplicate recall values results in $k$ recall/precision pairs where $k \neq m$. For each recall/precision pair, we then calculate the F-measure which is a weighted combination of recall and precision. We define F-measure as:

$$F = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

Thus, for each abstract in the testing set, we return $k$ recall/precision/F-measure values. The overall performance of a classification method is then calculated by taking the maximum F-measure of each abstract and averaging such maximum F-measures over all abstracts in the testing set, or

$$\hat{F} = \frac{1}{N} \sum (\max(F_1), \max(F_2), \max(F_3), ..., \max(F_n))$$

where $N =$ number of abstracts in the testing set

$F_x =$ set of all F-measures of the $x$th abstract in the testing set

4 Results

As mentioned above, parameter selection was undertaken using the tuning and tune testing data sets with the following eleven values for the parameter of number of classes: 10, 25, 50, 75, 100, 125, 150, 175, 200, 215, 224. Tables 1 and 2 below present the average F-measures for SVD and PLSI smoothing, respectively, on each of the data set collection (scz and p53). Figure 1 shows a graphical representation of the same data.

The tables indicate that the desirable values for C using SVD smoothing are 100 and 25 for collection scz and p53, respectively. For PLSI smoothing, the desirable values
<table>
<thead>
<tr>
<th>C (number of classes)</th>
<th>scz</th>
<th>p53</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.3702</td>
<td>0.4001</td>
</tr>
<tr>
<td>25</td>
<td>0.5105</td>
<td>0.4665</td>
</tr>
<tr>
<td>50</td>
<td>0.5519</td>
<td>0.4599</td>
</tr>
<tr>
<td>75</td>
<td>0.5594</td>
<td>0.4391</td>
</tr>
<tr>
<td>100</td>
<td>0.5614</td>
<td>0.4284</td>
</tr>
<tr>
<td>125</td>
<td>0.5391</td>
<td>0.4242</td>
</tr>
<tr>
<td>150</td>
<td>0.5355</td>
<td>0.4134</td>
</tr>
<tr>
<td>175</td>
<td>0.5183</td>
<td>0.4051</td>
</tr>
<tr>
<td>200</td>
<td>0.5219</td>
<td>0.4088</td>
</tr>
<tr>
<td>215</td>
<td>0.5146</td>
<td>0.4007</td>
</tr>
<tr>
<td>224</td>
<td>0.5184</td>
<td>0.3896</td>
</tr>
</tbody>
</table>

Table 1: Parameter Selection: Average F-measures for scz and p53 collections using Naive Bayes Classifier with SVD smoothing.

<table>
<thead>
<tr>
<th>C (number of classes)</th>
<th>scz</th>
<th>p53</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.4647</td>
<td>0.4207</td>
</tr>
<tr>
<td>25</td>
<td>0.5705</td>
<td>0.5409</td>
</tr>
<tr>
<td>50</td>
<td>0.6290</td>
<td>0.5839</td>
</tr>
<tr>
<td>75</td>
<td>0.6329</td>
<td>0.5854</td>
</tr>
<tr>
<td>100</td>
<td>0.6327</td>
<td>0.5940</td>
</tr>
<tr>
<td>125</td>
<td>0.6404</td>
<td>0.5882</td>
</tr>
<tr>
<td>150</td>
<td>0.6482</td>
<td>0.5836</td>
</tr>
<tr>
<td>175</td>
<td>0.6452</td>
<td>0.5906</td>
</tr>
<tr>
<td>200</td>
<td>0.6532</td>
<td>0.5943</td>
</tr>
<tr>
<td>215</td>
<td>0.6494</td>
<td>0.5904</td>
</tr>
<tr>
<td>224</td>
<td>0.6492</td>
<td>0.5976</td>
</tr>
</tbody>
</table>

Table 2: Parameter Selection: Average F-measures for scz and p53 collections using Naive Bayes Classifier with PLSI smoothing.
for C are 200 and 224 for collection scz and p53, respectively. It can also be seen that the optimal parameter value is uniquely determined by both the smoothing method employed and the “identity” of the data set collection and not by one alone.

<table>
<thead>
<tr>
<th>Smoothing Method</th>
<th>scz</th>
<th>p53</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVD</td>
<td>100</td>
<td>25</td>
</tr>
<tr>
<td>PLSI</td>
<td>200</td>
<td>224</td>
</tr>
</tbody>
</table>

Table 3: Parameter Selection: Optimal values for number of classes for SVD and PLSI smoothing, given each of the scz and p53 collection.

After parameter selection, we proceed with the main process of MeSH term classification using the training and testing data sets. The optimal parameter values obtained from the parameter selection process are used for SVD and PLSI smoothing. The remaining four classification methods are run as is, without any parameter values. Table 4 below shows the average F-measures for all six classification methods on each of the data set collection (scz and p53). Figure 2 shows a graphical representation of the same data.

In addition, we show below precision vs. recall graphs of the best run for each of the six classification methods. These precision vs. recall figures refer to a particular abstract (either in the scz or p53 collection) which was classified optimally by the stated classification method.
### Table 4: MeSH Term Classification: Average F-measures for scz and p53 collections using the six different classification methods.

<table>
<thead>
<tr>
<th>Classification Method</th>
<th>scz</th>
<th>p53</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Smoothing</td>
<td>0.4404</td>
<td>0.4189</td>
</tr>
<tr>
<td>LaPlace</td>
<td>0.1790</td>
<td>0.1938</td>
</tr>
<tr>
<td>SVD</td>
<td>0.5475</td>
<td>0.4712</td>
</tr>
<tr>
<td>PLSI</td>
<td>0.6326</td>
<td>0.6168</td>
</tr>
<tr>
<td>SOCR</td>
<td>0.0917</td>
<td>0.0838</td>
</tr>
<tr>
<td>KNN</td>
<td>0.6772</td>
<td>0.6154</td>
</tr>
</tbody>
</table>

Figure 2: Average F-measures for scz and p53 collections using the six different classification methods.
Figure 3: Precision-Recall Curves for Different Classification Methods
5 Conclusions

We show here that, despite attaining maximum likelihood on the training set, a maximum-likelihood estimator performs poorly for text classification tasks, due to overfitting. We present a number of probability estimators, and we show two main results:

1. One’s choice of estimator for $P(w)$ is very important to the generalization performance of your classifier on test data.

2. With the right choice of classifier (here, PLSI with an optimal number of classes) Naive Bayes can work as well as K-nearest neighbors, which has been shown to be one of the better performing classifiers.

The performance of these models in classification also hints at the validity of the probabilistic models that underly them, and suggest further research in both text classification, and probabilistic models of language.