SECOND DRAFT
MAY 25, 2010

Predicting Risk from Financial Reports with Supervised Topic Models

Neel Shah
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
neelshah@cmu.edu

Noah A. Smith
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
nasmith@cs.cmu.edu

Abstract

Forecasting from analysis of text corpora is an exciting research area, one that has potential for application to a variety of fields such as finance, medicine and consumer research. We apply techniques from Natural Language Processing (NLP) to classifying documents with class labels based on real-world continuous quantities associated with the forward-looking portion of the text's meaning. In particular, we study Financial Reports because of the presence of a large text corpus that is highly standardized and widely studied by financial analysts in industry. In conducting our analysis we use a class of generative probabilistic models known as Topic Models. In such a model, documents are a mixture of topics, where a topic is defined as a probability distribution over words. These models are interesting because they provide a simple probabilistic procedure for generating documents. Such a procedure can be inverted using standard statistical techniques, allowing us to infer a set of topics from which a particular document was generated. We then associate the inferred topic distributions with class labels based on real-world quantities such as company-level financial indicators for the classification task.

1 Introduction

1.1 Motivation

The context of much of what follows is structured after Kogan et. al. [1]. We extend their work by solving a similar problem with a different class of models. In particular, while they use Support Vector Regression (SVR) to predict real-world continuous quantities associated with a document's meaning, we use Multi-Class Supervised Latent Dirichlet Allocation (Multi-Class sLDA) to assign class labels based on the same real-world continuous quantities associated with the same document's meaning. Since Support Vector methods can be used for classification as well, uses and implementations of which are very well-documented, one of our earliest motivations was for this work to provide a side-by-side comparison of discriminative (Support Vector Classification) and generative (Multi-Class sLDA) classification methods applied to the same problem space.

1.2 Problem Statement

We are solving a text classification problem: given a piece of text, predict a class label based on a real-world continuous quantity associated with the text's meaning. In particular, we use a company's annual financial report to classify the financial risk of investment in that company, as measured empirically by a quantity known as *stock return volatility*.

Predicting financial risk is important to stock investors and financial analysts who use such company-level indicators for building an optimal portfolio and publishing company evaluations in secondary

reports, respectively. While these financial reports are by no means an exclusive part of an investor's or analyst's decision criterion, they are often the first thing considered when trying to build a holistic picture of a company's state of current operations and future profitability. The unique place they occupy in the financial community is because of the fact that they are government mandated (by the SEC) and anything included in such reports is required, by law, to be accurate. The assumption is that these financial reports contain a significant amount of information about a company's value. While [1] explored the secondary question of whether these costly-to-produce reports are actually informative and whether they serve their purpose of actually protecting the investor, we solely focus on our model's predictive performance on this classification task.

As hinted at, we are interested in solving this problem primarily because it is a test-bed for NLP research, specifically in the area of Text-Driven Forecasting. The output variable (volatility), empirically measured, is uncontroversial and widely used in the financial community in almost every discussion of a candidate stock's potential for investment. Unlike other tasks in Statistical NLP, prediction tasks usually have target variables whose usefulness is controversial. Because our output variable is a summary statistic about the real-world, it is independent of human expertise, knowledge or intuition. This is a large part of the reason why we extend the work of [1], because the "prediction task proves a new objective test-bed for any kind of linguistic analysis."

Unlike in other NLP problems, we do not have to rely on costly annotated resources. By law, both the text and the historical financial data are freely available as a byproduct of the American financial system. Additional data can be obtained almost effortlessly by anyone with the right text mining tools. For our research, we use the data collected by the authors of [1] for their work, additional details of which are outlined in the Section 2.

1.3 Notation and Terminology

We present standardized notation and language for modeling text collections. Core entities are "words," "documents," and "corpora." Topic models aim to introduce latent variables that represent abstract notions such as topics, which may not always correlate with our notion of topics of a document. Topic Models are not tied to text and many successful applications of the model we use have been documented in other problem areas, particularly image label classification as in [4].

The terms defined below (as in [3]) will be used in the sections that follow (unless otherwise specified):

- A *word* is the basic unit of text corpora, which we define to be an item from a vocabulary indexed by $\{1, ..., V\}$.
  Words are represented as unit-basis vectors that have a single component equal to one (corresponding to the index position in the vocabulary) and all other components equal to zero. Thus, using superscripts to denote components denote components of the vth word in the vocabulary is represented by a V-vector w such that $w^v = 1$ and $w^u = 0$ for $u \neq v$.
- A *document* is a sequence of N words denoted by $\mathbf{w} = (w_1, w_2, ..., w_N)$, where w_n is the nth word in the sequence.
- A *corpus* is a collection of M documents denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

2 Dataset

2.1 Financial Reports

Financial Reports or "Form 10-K" statements are produced by all publicly-traded companies as required by the Securities and Exchange Commission (SEC). As summarized in [1], each company-level report is in a standardized format and is intended to give a comprehensive summary of the company's performance. Each report typically contains historical data about the company's organization and financial data about its operations. These reports are publicly available and regularly published on the SEC's website[1]. The structure of the 10-K report also specified in detail on the SEC's website. The authors of [1] have collected 54,379 reports published over the ten-year period

[1] http://www.sec.gov/edgar.shtml

1996-2006 from 10,492 different companies. Since each report has a date of publication, we can tie the text back to the financial variables we want to predict.

With the goal of predicting future events in mind, we choose to focus on a specific section of the 10-K report. This is Section 7, also known as "Management's Discussion and Analysis" (MD&A) section. Within this section, we focus on Subsection 7A, also known as "Quantitative and Qualitative Disclosures about Market Risk." By law, companies have to put a disclaimer on forward-looking statements to the effect that projections of future performance are not guaranteed, and things could go otherwise. Because of this, most, if not all, forward-looking text statements are contained in Section 7 within Subsection 7A. All other sections are filtered from the report.

In [1], the filtering procedure is performed by the a lightweight hand-written Python script that does loose string matching for the Section 7, 7A and 8 headers. It finds the longest reasonable "Section 7" match (in words) of more than 1,000 whitespace-delineated tokens. Section 7, and the entire Report in more recent years, typically begins with an introduction as follows (from H&R Block's 2005 Form 10-K, before tokenization, for readability; boldface added):

> In this report, and from time to time throughout the year, we share our expectations for the Companys future performance. These **forward-looking statements are based upon current information, expectations, estimates and projections** regarding the Company, the industries and markets in which we operate, and our assumptions and beliefs at that time. **These statements** speak only as of the date on which they are made, **are not guarantees of future performance, and involve certain risks, uncertainties and assumptions**, which are difficult to predict. Therefore, actual outcomes and results could materially differ from what is expressed, implied or forecast in these forward-looking statements. Words such as believe, will, plan, expect, intend, estimate, approximate, and similar expressions may identify such forward-looking statements.

Note that some documents downloaded do not pass the filter at all and are excluded from the work in [1] and present work as well. For example, some reports that include Section 7 "by reference" are excluded by the filter because the text is not directly included in the document.

The authors of [1] tokenized the text, which included "punctuation removal, downcasing, collapsing all digit sequences, and heuristic removal of remnant markup." Since our goal is to focus on finding indicators of risk directly from the text, the removal of numerical information is justified. If we used the numerical information as predictors of risk we would be using financial data streams directly, which circumvents our goal of using the text reports.

Table 1 gives statistics for the corpora used in this work, which is a subset of the corpus without missing volatility measurements. The authors of [1] explain the drastic increase in length during the 2002-2003 period by the passage of the Sarbanes-Oxley Act of 2002 by the U.S. Congress (and related action by the SEC) in the wake of Enron's accounting scandal. The new regulation imposed revised standards on what publicly-traded companies in the U.S. should report.

year	words	documents	words/doc.
1996	5.5M	1,408	3,893
1997	9.3M	2,260	4,132
1998	11.8M	2,462	4,808
1999	14.5M	2,524	5,743
2000	13.4M	2,425	5,541
2001	15.4M	2,596	5,928
2002	22.7M	2,846	7,983
2003	35.3M	3,612	9,780
2004	38.9M	3,559	10,936
2005	41.9M	3,474	12,065
2006	38.8M	3,308	11,736
total	247.7M	26,806	9,240

Table 1: Dimensions of the dataset used in [1] and this work after filtering and tokenization.

2.2 Volatility Measurements

The financial community widely regards stock-return volatility as a measure of *risk*. By definition it is the standard deviation of a stock's return over a finite window of time. Stock-return volatility is directly related to the range of fluctuations in a stock's price: a stock has high volatility when its price fluctuates widely and low volatility when its price fluctuates narrowly or stays constant.

We repeat the derivation of stock-return volatility as the authors of [1] have.

Let $r_t = \frac{P_t}{P_t - 1} - 1$ be be the return on a given stock between the close of trading day $t-1$ and day t, where P_t is the(dividend-adjusted) closing stock price at date t. The measured volatility over the time period from day $t-1$ to day t is equal to the sample standard deviation:

$$v_{[t-\tau,t]} = \sqrt{\sum_{i=0}^{\tau} (r_{t-i} - \bar{r})^2 \Big/ \tau} \tag{1}$$

where r is the sample mean of r_t over the period. In [1], the above estimate is treated as the true output variable on training and testing data.

However, note that this is not the only the volatility measurement available to us. Another popular measure of volatility usually treated with the same level of attention as stock-return volatility is implied volatility. Implied volatility assumes that the observable stock returns come from a model of the stock's price at any given a time. For example, such a model could assume the stock price follows some continuous-time stochastic process dependent on a fixed set of parameters, one of which is stock volatility. Given a set of stock price observations over a given period of time and all the model parameters except for stock volatility, one can use the model to derive the volatility *implied* by the stock price observations, values of the other fixed parameters and the model's assumptions. Calculating implied volatility is as easy as calculating stock-return volatility because of the widely documented models for stock-price movements and inversion procedures that have been produced for them. One cannot help but notice the parallels between our generative model and the procedure for obtaining implied volatility. While this approach allows us to encode assumptions about a particular market's stock price movement, its reliance on financial expert knowledge makes it a subjective output variable. However, we want our target to be based on observation rather than theory in order for our prediction task to remain extensible to other forms of linguistic analysis. Thus, we maintain our choice of stock-return volatility as our output variable.

As in [1], we note the differences between predicting stock-returns volatility and predicting stock-returns. Both are fundamentally different tasks. In the former, we are interested in predicting how stable a stock's price will be over a future time period. In the latter, we are predicting how well a stock will perform. As is acknowledged by the financial community, directly predicting a stock's performance based on easily available public information is difficult due to the "efficient market hypothesis" [9]. In contrast, predicting a stock's riskiness using public information is uncontroversial and an underlying assumption in many economically sound pricing models.

If the "efficient market hypothesis" is to be believed, as is empirically suggested, predictability of returns, if possible, could be traded away by virtue of buying/selling stocks that are under- or over-valued [9]. In short, predicting stock returns is based on the principle that if it could have been done, it would have been done already and any gains from doing so would have been traded away. On the other hand, a similar strategy costs much more to implement using predictability of volatility.

For each report included in our corpus, the authors of [1] used the Center for Research in Security Prices (CRSP) US Stocks Database to obtain the price return series along with other firm characteristics. Using the above definition of stock-return volatility, they calculated two volatilities for each company/report observation: the twelve months prior to the report ($v^{(-12)}$) and the twelve months after the report ($v^{(+12)}$).

Since we are solving a classification problem, the real-valued continuous volatility measurements are converted into volatility class labels. For a given corpus, we gather volatility measurements for all documents in the corpus as above and store them in a list of size n. Then we sort this list and divide it into k partitions where k is the number of class labels we desire. Then we go over each

partition i (starting with the lowest, or $i = 0$) where $i \in \{0, ..., k-1\}$ and assign the label i to all the associated documents in that partition.

For example, if $k = 2$, we want 2 class labels, then $i \in \{0, 1\}$, which corresponds to {low, high} volatility labels. We use the median of the sorted list, to divide the members of the list into two partitions. For volatility measurements less than the median, we label the associated documents with the label 0 or "low volatility" and similarly, for volatility measurements greater than the median, we label the associated documents with the label 1 or "high volatility." This procedure is readily extensible to higher k.

In order to extend to higher k, we first assume that the sorted list of volatility measurements has size n, such that n is divisible by k. When this is the case, the division points for the partitioning algorithm are at $\frac{m*n}{k}$ where $m \in \{1, ..., k-1\}$. This is based on the assumption that the size of the sorted list n is divisible by k. When n is not divisible by k, we set $n = n - (n \bmod k)$. What this effectively does is (a) make the new n divisible by k and also (b) assigns the highest class label $k-1$ to the remainder $n \bmod k$) associated documents at the end of the sorted list. Since there are at most $k-1$ of these remainder associated documents, and k generally tends to remain small (i.e. $k \ll 10$) and n generally tends to remain large (i.e. $n \gg 1000$), we are justified in doing this.

Transforming real-world measurable continuous quantities to class labels in such a way naturally induces an ordering on the labels. This makes our volatility class labels ordinal, or ordered, instead of nominal, or unordered. Ordinal labels allow you rank labels on a scale, but the real distance between categories is unknown. For nominal values the real distance between categories is assumed to be a uniform fixed constant between any two labels, or in other words, you cannot rank labels on a scale. This may have a significant impact on Multi-Class sLDA, which has been documented to show better than state-of-the-art performance only on nominal, and piecewise independent, labels such as image classes as in [4]. Also, when picking our baseline, we have to choose an appropriate model from the family of Generalized Linear Models (GLMs) for ordinal categorical data [10].

3 Models and Algorithms

3.1 Elastic-Net Multinomial Regression

We use Elastic-Net Multinomial Regression as our Baseline.

The authors of [11] developed a technique to improve the performance of multinomial regression. This regularization technique is called **Elastic Net** and it simultaneously performs variable selection and continuous shrinkage. The naive method is a least squares method with an l_1 penalty and a quadratic penalty. The l_1 penalty, related to lasso-type thresholding, performs variable selection and induces a sparse model. The quadratic penalty, related to ride regression, places no limitation on the number of variables that may be selected for the model and induces a grouping effect. The elastic net procedure is a scaled transformation of the naive method, retaining the variable selection property while correct for additional bias, without reducing variance, introduced by extra shrinkage. For full details, which are out of the scope of this work, see [11].

We are interested in this technique because it is widely regarded as a state of the art discriminative classification model. We will apply the technique to the sets on which Multi-Class sLDA is applied in order to evaluate the performance of our work.

3.2 Latent Dirichlet Allocation (LDA)

Developed by authors of [2], LDA is a generative probabilistic model, specifically, a three-level hierarchical Bayesian model, for text corpora. Each document is modeled as a finite mixture over an underlying set of topics[2]. Each topic is modeled as an infinite mixture over an underlying set of topic probabilities. The topic probabilities provide an explicit representation of the document. Topics can be said to represent an underlying semantic theme; a document with a large number of words can be modeled as being composed from a smaller number of topics [3].

[2]latent multinomial variables that representation probability distributions on sets of words.

The authors of [2] present efficient approximate inference techniques based on variational methods and an EM algorithm for empirical Bayes parameter estimation. In the rest of the section we summarize the finer points of [2] and its applicability to our problem.

The basic idea behind LDA is motivated by many of the advances in dimensionality reduction techniques. In the field of Information Retrieval (IR), the popular *tf-idf* scheme is used to reduce documents of arbitrary length to fixed-length lists of numbers. Further compression of large text corpora can be achieved by *latent semantic indexing* (LSI), which uses singular value decomposition to identify a linear subspace in the space of *tf-idf* features that captures most of the variance in the corpora. Probabilistic LSI, a generative model, was then created as an alternative to LSI. As opposed to LSI, which uses linear algebra to reduce dimensionality, pLSI models each word in a document as a sample from a mixture model, where mixture components are multinomial random variables and can be regarded as topics. In pLSI each document is represented as a list of mixing proportions of topics, leading to problems with assigning probability to a document outside the training set.

LDA aims to improve on pLSI, by using its underlying assumptions and by modeling at the document-level. LDA is based on the *bag of words* assumption, which states that the order of words in a document does not matter. This implies that the ordering of documents in a corpus does not matter as well. While pLSI used a mixture model based on the exchangeability of words, LDA considers mixture models based on the exchangeability of the words and the documents.

The unsupervised model aims to find a probabilistic model of a corpus that assigns high probability to members of the corpus and high probability to other "similar" documents.

LDA assumes the following generative process for each document $\mathbf{w}$ in a corpus D from [2]:

1. Choose $N \sim \text{Poisson}(\xi)$
2. Choose $\Theta \sim \text{Dir}(\alpha)$
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\Theta)$
   (b) Choose a word w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Simplifying assumptions from [2]:

1. Dimensionality k of the Dirichlet distribution and dimensionality of the topic variable z is assumed known and fixed.
2. Word probabilities are parameterized by a k x V matrix β where $\beta_{ij} = p(w^j = 1|z_i = 1)$, which is treated as a fixed quantity to be estimated.
3. Poisson assumption is not critical, and more realistic document length distributions can be used as needed.
4. N is independent of all other data generative variables (Θ and $\mathbf{z}$) and its randomness can be ignored.

The k-dimensional Dirichlet random variable Θ can take values in the $(k-1)$-simplex (a k-vector Θ lies in the $(k-1)$-simplex if $\Theta_i \geq 0, \sum i = 1^k \Theta_i = 1$). The Dirichlet distribution on the simplex is in the exponential family, has finite dimensional sufficient statistics, and is conjugate to the multinomial distribution. Given the parameters α and β the joint distribution of a topic mixture Θ, a set of N topics $\mathbf{z}$, and a set of N words $\mathbf{w}$ can be found. As is shown in [2], this can be used to obtain the marginal distribution of a document, which can be then used to obtain the probability of a corpus.

The probabilistic graphical model of LDA in Figure 1 displays its three-levels. The parameters α and β are corpus-level parameters, assumed to be sampled once in the process of generating a corpus. The variables Θ_d are document-level parameters, sampled once per document. Finally, the variables z_{dn} and w_{dn} are word-level variables and are sampled once for each word in each document. The key difference between LDA and the simple Dirichlet-multinomial clustering model is that in LDA, documents can be associated with multiple topics.

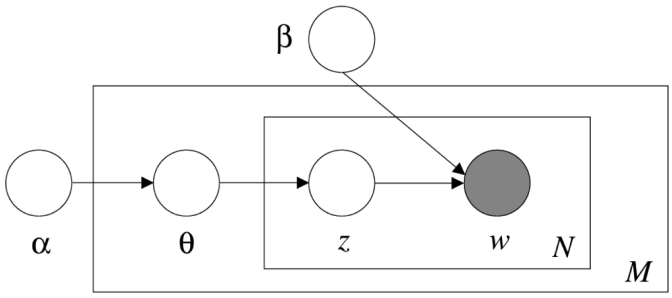

Figure 1: Graphical model representation of LDA. The boxes or "plates" represent replicates. From outside of outer plate to inside of inner plate: corpus level → document level → word level

Variational Inference

From [2], LDA assumes that each word of both the observed and unseen documents is generated by a randomly chosen topic which is drawn from a distribution with a randomly chosen parameter. This parameter is sampled once per document from a smooth distribution on the topic simplex. Such a model is a *parametric empirical Bayes model*, for which we can use the empirical Bayes approach to estimating parameters such as α and β in simple implementations of LDA.

In order to solve the inference problem for LDA, one needs to compute the posterior distribution of the hidden variables given a document. However, this distribution is intractable for exact inference. The authors of [2] use a simple convexity-based variational algorithm for inference in LDA. The derivation of the variational inference procedure, fully outlined in [2], is out of the scope of this work, but we summarize the variational inference algorithm from [2] in Figure 2.

1. initialize $\phi^0_{ni} := 1/k$ for all i and n
2. initialize $\gamma_i := \alpha_i + N/k$ for all i
3. **repeat**
4. **for** $n = 1$ to N
5. **for** $i = 1$ to k
6. $\phi^{t+1}_{ni} := \beta_{iw_n} \exp(\psi\gamma^t_i)$
7. normalize ϕ^{t+1}_n to sum to 1.
8. $\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi^{t+1}_n$
9. **until** convergence

Figure 2: A variational inference algorithm for LDA

Each iteration of the variational inference for LDA requires only $O((N+1)k)$ operations [2]. In practice the number of iterations required for a single document is on the order of the number of words in the document, bounding the total number of operations to the order N^2k

Parameter Estimation

The empirical Bayes method is used for parameter estimation in the LDA model. Given a corpus of documents $D = \{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_M}\}$, the authors of [2] find parameters α and β that maximize the (marginal) log likelihood of the data:

$$l(\alpha, \beta) = \sum_{d=1}^{M} \log \mathrm{p}(w_d|\alpha, \beta) \tag{2}$$

The likelihood function contains the intractable quantity $p(\mathbf{w}|\alpha, \beta)$. They use the variational inference procedure to obtain a lower bound on the log likelihood, which can be maximized with respect to α and β.

The approximate empirical Bayes estimates are calculated via an alternating *variational EM* procedure that maximizes a lower bound with respect to the variational parameters γ and ϕ and then for fixed values of the variational parameters, maximizes the lower bound with respect to the model parameters α and β.

1. (E-step) For each document, find the optimizing values of the variational parameters $\{\gamma_d^*, \phi_d^* : d \in D\}$. This is done as is described in the Variational Inference subsection.
2. (M-step) Maximize the resulting lower bound on the log likelihood with respect to the model parameters α and β. This corresponds to finding the maximum likelihood estimates with expected sufficient statistics for each document under the approximate posterior which is computed in the E-step.

Figure 3: A variational EM procedure for LDA

We summarize the iterative algorithm from [2] in Figure 3. The two steps below are repeated until the lower bound on the log likelihood converges. More efficient methods for parameter estimation are documented in [2], which are once again out of scope for this work.

Applicability to our Work

LDA, in its unsupervised form, can be used to model the documents in our corpus. We could potentially use the generated topics as features for a logistic regression model, to predict volatility measurements or class labels. However, as we will see later, extending LDA to the supervised setting allows us to train topics based on which are most predictive of a particular response. As opposed to the vanilla LDA model, Supervised LDA (sLDA) actually allows us to pair documents and response variable measurements or class labels. Using the topics along with the words in a corpus as features in a logistic regression model has been preliminary tested, but was found to be no more helpful than using words (or N-grams) alone. Also, predictive performance deteriorates drastically when topics alone are used as features.

Another potential approach to document classification with the unsupervised model was demonstrated in [2]. The documents in the corpus were reduced to a fixed set of real-valued features - the posterior Dirichlet parameters $\gamma^*(\mathbf{w})$. This was primarily done in order to test how much discriminatory information was lost in reducing the document description to these parameters. These low dimensional representations of the documents in a corpus were used to train Support Vector Machines (SVM) and it was found that while there was a significant reduction in the feature space, there was little reduction in classification performance using LDA-based features. We have yet to try this approach using our data, which would shift our focus from using a Supervised Topic Model to directly predict the class label instead to using an Unsupervised Topic Model to reduce dimensionality so that discriminative models such as SVM would be used for the prediction task. However, experiments in that direction would definitely help us compare LDA's use as a tool for dimensionality reduction versus a tool for regression or classification.

Document modeling for the purposes of qualitatively evaluating predicted topics seems like a promising task. The results of such experiments are outlined in Section 5. We chose to focus on predicting topics from the training and test sets that were used in the Supervised Learning task. This was in order to illustrate the differences in the top-topic words chosen by each model and empirically confirm the different goals of the LDA and sLDA models.

3.3 Supervised Topic Models (sLDA)

The authors of [3] extended the work from [2] to develop sLDA, a statistical model of labelled documents that accommodates a variety of response types. In their work they derive an approximate maximum-likelihood procedure for parameter estimation, which relies on variational methods to handle intractable posterior expectations. Their primary goal is to use the fitted model to predict

response values for new documents. Note that the authors of [3] primarily focus on solving regression problems, which was motivated the growing need to analyze large text corpora, especially in the case when documents can be easily paired with an external response variable.

Similar to Section 3.2, in the rest of this section we summarize the finer points of [3] and its applicability to our problem.

sLDA adds to LDA a response variable connected to each document. This response variable is a quantity we are interested in predicting, such as the volatility measurements in [1]. The documents and responses are jointly modeled to find the hidden topics that will best predict response variables for future unlabeled documents. sLDA in [3] uses the same probabilistic methods as a generalized linear model to allow for various response types: unconstrained real values, real values constrained to be positive, unordered or **ordered class labels**, nonnegative integers and other types.

Fix the model parameters: K topics, $\beta_{1:K}$ (each β_k is a vector of term probabilities), a Dirichlet parameter α and response parameters η and δ.

sLDA assumes the following generative process for each document and response from [3]:

1. Draw topic proportions $\Theta|\alpha \sim \text{Dir}(\alpha)$
2. For each word
   (a) Draw topic assignment $z_n|\theta \sim \text{Mult}(\theta)$
   (b) Draw word $w_n|z_n, \beta_{1:K} \sim \text{Mult}(\beta_{z_n})$
3. Draw response variable $y|z_{1:N}, \eta, \delta \sim \text{GLM}((\bar{z}), \eta, \delta)$ where we define
   (1) $\bar{z} := (1/N)\sum_{n=1}^{N} z_n.$

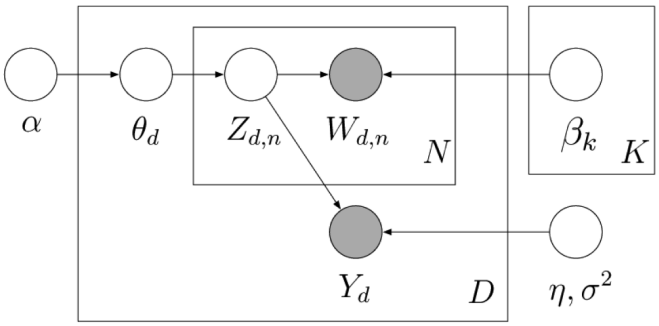

Figure 4: Graphical model representation of sLDA. The boxes or "plates" represent replicates.

The distribution of the response is a generalized linear model, which contains a "systematic component" and a "random component," assumed to be from an exponential family. As opposed to the words and the documents, the responses are treated as non-exchangeable and response is treated as it depends on the topic frequencies which actually occur in a generated document. Figure 4 summarizes the parameters that need to be estimated. In the case of sLDA, β, previously our corpus-level parameter, is sampled once for each topic.

3.3.1 Variational Inference

Like for LDA, when attempting to solve the posterior inference problem by computing the conditional distribution of the latent variables at the document-level given its words $w_{1:N}$ and the corpus-wide model parameters, they arrive at a conditional distribution that is intractable.

The authors of [3] use mean-field variational inference, where Jensen's inequality is used to lower bound the normalizing value. The extended derivation is found in their work and is not presented here because its complexity is out of our scope. They key difference between sLDA and LDa is in the update for the variational multinomial ϕ_n. Since the optimization with respect to the variational multinomial depends on the form of the partial derivative with respect to ϕ_n of the expectation of the log-normalizer, it is dependent on our choice of the distribution of response type. This problem will be revisited in the inference procedure for Multi-Class sLDA, in Section 3.4.1.

3.3.2 Parameter Estimation

Maximum likelihood estimation based on variational expectation-maximization is used to estimate the Dirichlet parameters α, GLM parameters η and δ and topic multinomials $\beta_{1:K}$ from a data set of observed document-response pairs $\{w_{d,1:N}, y_d\}_{d=1}^{D}$. From [3], the expectation is taken with respect to the variational distribution. The maximization proceeds by maximum likelihood estimation under expected sufficient statistics.

Variational EM optimizes corpus-level lower bound on the log likelihood of the data. Response variable's y are augmented with document indices to make y_d. Similarly, empirical topic assignment frequencies $\bar{Z}$ are augmented to make $\bar{Z}_d$ and so on. Expectations are taken with respect to document-specific variational distributions $q_d(z_{1:N,\Theta})$.

1. (E-step) Estimate the approximate posterior distribution for each document-response pair using the variational inference algorithm from Section 3.3.1, fully developed in [3].
2. (M-step) Maximize the corpus-level *evidence lower bound* or ELBO with respect to the model parameters. More details on the ELBO are provided in the development of the variational inference algorithm in [3].

Figure 5: A variational EM procedure for sLDA

Variational EM finds a local optimum of the likelihood function of the model parameters. The M-step updates of the topics $B_{1:K}$ are the same as in LDA. The procedure for estimating the GLM parameters are once again response distribution choice dependent and will be covered in Section 3.4.2. Since we fix the Dirichlet parameter $\alpha = 1/K$ where K is the number of topics, an input to the model, we do not need to estimate it for our problem.

3.3.3 Prediction

Given a newly observed document $w_{1:N}$ and the fixed values of the model parameters as a fitted model $\alpha, \beta_{1:K}, \eta, \delta$, we are interested in predicting a response y or the expected response value. This step depends on approximating the posterior mean of $\bar{Z}$ using variational inference, which we discussed briefly in Section 3.3.1. The procedure is the same as in that section, but the terms depending on the response y are removed from the ELBO. The authors of [3] use a coordinate ascent algorithm identical to variational inference for LDA. This algorithm is independent of the particular response type. In summary, given a new document, we first compute the variational posterior distribution of the latent variables Θ and Z_n or $q(\theta, z_{1:N})$ and then estimate the response by computing or approximating $E_q[\mu(\eta^T \bar{Z})]$ where $\mu(.) = E_{GLM}[Y|.]$. This quantity is calculated as a part of the estimating the GLM parameters, discussed briefly, in the Section 3.3.2.

3.3.4 Applicability to our Work

We started with a presentation of sLDA from [3] first because it was the basis for the development of Multi-Class sLDA. While the specific algorithms developed are for a Gaussian and a Poisson response, the inference and estimation methods are suggestive of extensions to any other exponential family. Since our classification task requires a Multinomial response, a member of the exponential family, the work in [3] is very relevant to accomplishing the goals of this work. Specifically, it is the assumption on the distribution of the response as a GLM gives us the flexibility to model any response type from the exponential family, particularly the multinomial response type, by specifying the base measure and log-normalizer [3].

For general exponential family response, the authors of [3] recommend using the multivariate delta method for moments to approximate difficult expectations as been documented in their work to be effective in variational approximations.

3.4 Multi-Class sLDA

The authors of [4] extend the work from [3] to develop Multi-Class sLDA, a probabilistic model that simultaneously learns the latent topics among the documents that are predictive of their class labels. For new unlabeled documents, the model provides predictive distributions of the class label.

Multi-Class sLDA assumes the following generative process for each document and class label:

1. Draw topic proportions $\Theta \sim \text{Dir}(\alpha)$
2. For each word
   (a) Draw topic assignment $z_n|\theta \sim \text{Mult}(\theta)$
   (b) Draw word $w_n|z_n \sim \text{Mult}(\beta_{z_n})$
3. Draw class label $c\big|z_{1:N} \sim \text{softmax}(\bar{z}, \eta)$, where $\bar{z} = \frac{1}{N}\sum_{n=1}^{N} z_n$ is the empirical topic frequencies and the softmax function provides the following distribution,
$$p(c|\bar{z}, \eta) = \exp(\eta_c^T \bar{z}) \Big/ \sum_{l=1}^{C} \exp(\eta_l^T \bar{z})$$

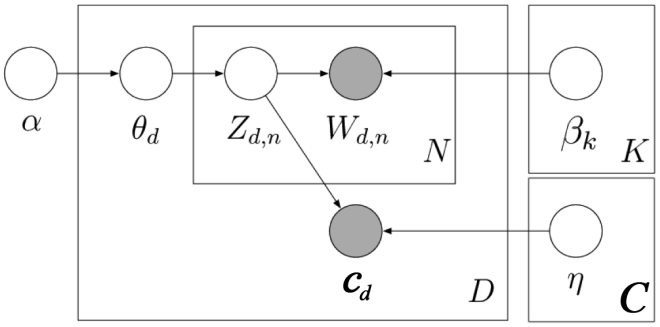

Figure 6: Graphical model representation of Multi-Class sLDA. The boxes or "plates" represent replicates.

The graphical model of Multi-Class sLDA displays the parameters that need to be estimated. Note the similarity between this and Figure 4 (changes are bolded). In sLDA, the response variable for each document is assumed drawn from a generalized linear model with input given by empirical distribution of topics that generated the words. In [3], the response variable is real valued and drawn from a linear regression. Since our goal is to build a classifier,we consider a class label response variable, drawn from a softmax regression for classification. This complicates the approximate inference and parameter estimation algorithm, but extends the work done by authors of [3].

3.4.1 Variational Inference

As we discovered was the case for LDA and sLDA, the posterior inference is not directly possible. Mean-field variational methods for a scalable approximation algorithm are applied. The full derivation is in [4].

3.4.2 Parameter Estimation

Provided a corpus of documents with class labels, $D = \{(w_d, c_d)\}_{d=1}^{D}$, we find the maximum likelihood estimation for text topics $\beta_{1:K}$ and class coefficients $\eta_{1:C}$. As in LDA and sLDA, we use the variational EM, which replaces the E-step of expectation maximization with variational inference to find an approximate posterior for each data point. In the M-step, as in exact EM, we find approximate maximum likelihood estimates of parameters using expected sufficient statistics calculated in the E-step.

3.4.3 Prediction

Prediction involves classification of unlabeled documents. First we need to perform variational inference given the unknown document. Use a variant of the algorithm in 3.4.1 to determine $q(\theta, z)$. Because the class label is not observed we remove the λ_{mn} terms from the variational distribution and the terms involving η_c from the updates on the topic multinomials.

From [4], the probability of the label c is estimated by replacing the posterior $\mathrm{p}(z|w,r)$ with the variational approximation

$$p(c\,|\,r,w) \approx \int \exp(\eta_c^T \bar{z} - \log(\sum_{l=1}^{C} \exp(\eta_l^T \bar{z})))q(z)dz$$
$$\geq \exp$$

where the last equation is obtained using Jensen's inequality and q is the variational posterior computed in the first step.The second term in the exponent is constant with respect to class label. Thus the prediction rule is

$$c* = \arg\max_{c\in 1,\ldots,C} E_q[\eta_c^T \bar{z}] = \arg\max_{c\in 1,\ldots,C} \eta_c^T \bar{\phi}$$

We apply two approximations. First we approximate the posterior with q. Second we approximate the expectation of an exponential using Jensen's inequality. This could be a potential source of error, because such approximations, though supported by [4] are admitted to be theoretically unfounded.

3.4.4 Applicability to our Work

The model finds a set of topics that are predictive of class labels. This is precisely the problem we set out to find and use in our work.

4 Experimental Methodology

In this section, we describe how experiments were designed. In particular we focus on the construction of the tools, algorithms and training/test sets. The smallest unit we use to compose training/test sets is a year's worth of documents where the year $\in \{1996, ..., 2006\}$. From collections of these smaller units, or subsets of the set $\in \{1996, ..., 2006\}$, we construct larger training/test sets. Since we have the goal of predicting future volatility in mind, the units used in the construction of the training set are from years before the units used in the construction of the test set.

4.1 Data Format

Both LDA and Multi-Class sLDA implementations require the data to be in a specific format, one that reminds us of the *exchangeability* assumptions (for words and documents) that both models were build upon. This format, referred to as the LDA format, is as follows:

Each document is succinctly represented as a sparse vector of word counts. The data is a file where each line is of the form:

[N] [term1]:[count] [term2]:[count] ... [termN]:[count]

where [N] is the number of unique terms in the document, and the [count] associated with each term is how many times that term appeared in the document. Note that [term1] is an integer which indexes the term; it is not a string.

Each unique term in a document is given the distinction **type**. On the other hand, each instance of a particular unique term is given the distinction **token**.

The LDA format allows for a sparse and compact representation of our text corpora.

The original corpus is a collection of text documents. Each document contains the target section of the 10-K statement in words, space separated, for a particular company in a particular year.

The corpus in LDA format is a data file and a vocabulary file.

data file contains documents (newline delimited) in the format specified above (no original identifying text, just indices and counts)

vocabulary file contains the unique words that appear in the corpus, newline separated, with the line number (0 indexed) corresponding to number that is used as an index for the word in the data file.

The vocabulary file is only used to obtain the top-topic words and is not used by the estimation/inference procedures.

From the original corpus of documents, it is straightforward to convert into LDA format. Given the text of a particular document, we need to loop over each word in the document (space delimited), adding each word to our dictionary (a map) of <word> and <word counts>. We create such a dictionary for each document in the corpus. The intersection of all the words the appear in these dictionaries is the unique vocabulary of this corpus, which we constructed in the process of forming each individual dictionary. We sort the words in this vocabulary in alphabetical order and then write each word to the vocabulary file (newline delimited), thus forming the index of words to numbers to be used by the data files. Then for each dictionary (order of documents does not matter), we append to a data file in the format specified above. Once this has been done for all dictionaries, we have the data file in LDA format as our representation of our text corpus.

4.2 Filtering Procedure

Due to limitations in the Multi-Class sLDA implementation, we needed to employ a filtering procedure when designing training and test sets. The filtering procedure serves two main purposes. First, it ensures that the construction of the test set excludes any new words not encountered when constructing the training set. Second, it limits the number of total tokens of a corpus when forming training or test sets for input into the model.

The first concern has a simple fix. We begin by constructing a training set, which gives us a data file and a vocabulary file. Then using the training set's vocabulary file as a guest list during the construction of the test set, which excludes any words from the test corpus not found on the list. The reason why we need to do this is because the Multi-Class sLDA model works in two phases: estimation and inference. The inference procedure, which uses the fitted model to do prediction on the test set, cannot react to words not in the fitted model. The fitted model is constructed by the estimation procedure, which uses the training set to generate the fitted model. Because of this, the inference procedure cannot react to words not in the training set, which we exclude from our test set. We need to implement smoothing in the inference procedure if we want it to be able to handle new words (unseen during training), which is a more involved process.

The second concern has a more complex fix. We want to limit the total tokens of a corpus when forming training or test sets. The tokens of a corpus are the individual instances of words that appear in the corpus. In the results section we outline the pre-filter number of tokens and the post-filter number of tokens in order to illustrate the consequences of the procedure. However, we do not know beforehand the maximum number of total tokens in a training/test set the Multi-Class sLDA implementation can handle. Also, since tokens correspond to the individual instances of types appearing in the corpus, even if we did know an upper limit on the tokens, we have no way of choosing which ones to exclude. For example, there could be $>$ 10,000 tokens of the type "a" and "the", which is the amount we need to reduce our total token amount by, and we do not have a heuristic to decide between the two.

The basic idea is to eliminate tokens based on the least/most common types. For a particular type, such as the word "a" and "the", we know the exact number of times it occurs. Since the amount of occurrences of either type is large relative to the other types in the corpus, we know that these are more frequently occurring types. In this case, we would eliminate the more frequently occurring type between the two choices if we needed to meet a particular token target.

Our filtering procedure abstracts this idea further. We take as input a value, α, that is between 0 and 50. This variable lets us specify the level of filtering (as a percentage) that should occur, with higher numbers corresponding to higher levels of filtering. Given a map of <types> to the <count of instances of that type that occur in a corpus> we sort by the count. We now have a sorted list of length X with the least to most frequent types found in a corpus. We set the $\{$low, high$\}$ thresholds for filtering by taking the members of this sorted list at the $\alpha \times X$ and $(100 - \alpha) \times X$ positions. Any types below or above (exclusive) these $\{$low, high$\}$ thresholds are stop listed. For example, if we set α at 50, the $\{$low, high$\}$ thresholds would be set by the median member of the sorted list, effectively filtering all types from the corpus (except for the median type). Also, by setting α at 0, the $\{$low, high$\}$ thresholds would be the first and last elements of the sorted list, which corresponds to no filtering at all.

We typically set α at 1% or 5% depending on the number of documents in the corpus and the pre-filter token count. Depending on the contents of documents in a corpus, and the resulting pst-filter token count, this is adjusted. This filtering occurs on the training and test sets, however, it is typically not required on the latter. Since each document has a different type distribution we cannot pick a particular level α beforehand. We have to make judgements based on the size of the target corpus and whether test runs of Multi-Class sLDA fail. This is a very subjective process, however, we have tried to retain as many of the original tokens as we could have.

4.3 Baseline

We use the **glmnet** package and function in R published by the authors of [5] to create a baseline for Multi-Class sLDA.

The function works well with very large sparse data matrices to fit a regularization path for the Elastic Net for a multinomial regression model. The algorithm uses cyclical coordinate descent in a pathwise fashion. For further details about the implementation of this procedure, see [5].

We use the notation of Section 1.3, when describing what we provide as input to the glmnet function, with the following additions. Remember we are given that a *corpus* is a collection of M documents, a *document* is a sequence of N words and a *word* is an item indexed from a vocabulary of size V.

We create a sparse matrix P with dimensions $M \times V$. Each row of P corresponds to a document and each column corresponds to a unique word from the vocabulary. Each cell P_{ij} corresponds to the number of times word j (indexed in vocabulary) appears in document i. Given that the data is in LDA format, see Section 4.1, it is relatively easy to construct the matrix P.

We create another matrix R with the dimensions $M \times 1$. Since each row of P corresponds to a particular document, for that document, we populate the corresponding row of R with the response label associated with that document.

For a given corpus, we construct the sparse matrix P and take it to be the predictors in our glmnet function. Correspondingly, the matrix R is taken to be the response in our glmnet function. All that remains to be tweaked are the Elastic Net mixing parameter, α, which we fix at $\alpha = 0.01$ and maximum number of iterations we would like the fitting procedure to run, we set maxit $= 1000$.

4.4 Document Modeling

For LDA, we use C source code released by authors of [2] as is to perform document modeling.

4.5 Classification

For Multi-Class sLDA, we use C++ source code released by authors of [4] to run our experiments with slight modifications to handle varying training/test set size.

5 Selected Results

Table 2 describes the characteristics of the Training Sets, to which we assign an Index for reference in the next section. In particular we list the Training Year(s) in the set, Total Number of Documents, Pre-Filter and Post-Filter total token counts and total number of Stop types.

Table 3 describes the characteristics of the Test Sets, on which we use the model fitted on the Training Sets in Table 2 to do inference. In particular we list the Testing Year in the set, Training Set index, average Accuracy of Multi-Class sLDA and Baseline accuracy.

Index	Train Year	Docs	Pre-Filter	Post-Filter	Stop	α %
1	2001	2597	15,519,607	2,960,334	581	1
2	2002	2846	22,830,558	3,963,181	710	1
3	2003	3612	35,402,868	5,282,014	912	1
4	2004	3559	38,975,123	5,766,147	974	1
5	2005	3475	41,901,864	6,071,990	1036	1
6	2004,2005	7034	80,876,987	9,862,448	1269	1
7	2003,2004,2005	10,646	116,279,855	2,517,774	7296	5
8	2002,2003,2004,2005	13,492	139,110,413	2,753,075	7902	5
9	2001,2002,2003,2004,2005	16,089	154,630,020	2,885,818	8361	5

Table 2: Characteristics of training sets used in experiments.

Test Year	Index	Docs	Accuracy %	Baseline %
2006	1	3306	53.3	62
2006	2	3306	56.2	66.7
2006	3	3306	66.3	73.7
2006	4	3306	68	80.3
2006	5	3306	67.7	79.6
2006	6	3306	67.1	75.1
2006	7	3306	57.3	68.5
2006	8	3306	59.7	67
2006	9	3306	57.4	64.9

Table 3: Characteristics of test sets used in experiments.

6 Discussion

The driving force behind this work was to extend the results of [1] and develop benchmarks for a new direction in Text-Driven Forecasting. We have used a relatively new class of generative probabilistic models, Supervised Topic Models, in order to solve a *text classification* problem, in which we used the text to make predictions about volatility class labels that correspond to measurable real-world continuous quantities. We applied the technique to predicting financial volatility class label from companies' 10-K reports and found the initial average classification accuracy results to be promising. An extension would be to use Supervised Topic Models to solve the *texts regression* problem and use the text to directly make predictions about real-world measurable quantities as in [1]. Solving the text regression problem using Supervised Topic Models is not as easy as growing the number of class labels k in our Multi-Class sLDA model to approach ∞; instead, the target variable's distribution, the inference and estimation algorithms need to be modified as is done in [3]. Another way to make our work directly comparable to [1] is to train and test a Support Vector Classifier (SVC) on the same sets as documented in the Results Section. This is much easier to do since fast algorithms for SVC already exist. Another issue is the predictive performance of our model, which is better than random, however, still far from the discriminative baseline (Elastic-Net Logistic Regression). This raises questions as to the applicability of generative models as a whole to such prediction tasks, questions which we explore in the next Section.

7 Further Work

7.1 Discriminative vs. Generative Classifiers

As mentioned earlier, one of our first motivations was to contrast discriminative and generative classifiers in the same problem space. Further work needs to be dedicated to fully exploring this problem, however, we briefly comment on the high-level differences between the models using in [1] and in our work. The below definitions are extrapolated from [6].

Discriminative models (i.e. Support Vector Machines and Logistic Regression) model the dependence of an unobserved variable y on an observed variable x. This is accomplished by modeling the conditional probability distribution $P(y|x)$ and then using it to predict y from x. In contrast with generative models, they do not allow one to sample from the joint probability distribution $P(y, x)$.

Generative models (i.e. Latent Dirichlet Allocation and Naive Bayes) model data directly, which is treated as randomly generated observable data, given some hidden parameters. They specify the joint probability distribution $P(y, x)$ over observations x and labels y, from which the conditional probability distribution $P(y|x)$ can be formed using Baye's rule. In contrast with discriminative models, generative models can be used to generate values of any variable in the model, as opposed to only being able to sample target variables, y, conditioned on observed quantities, x.

The authors of [7] compare the predictive performance of models from the two types (Logistic Regression vs. Naive Bayes) to see if a difference between the two exists and what that difference is founded upon. We summarize some of their conclusions here and comment briefly on how it affects us. According to them, it is widely conceived that discriminative classifiers (which model the posterior $P(y|x)$ directly) almost always have a higher test set accuracy than generative classifiers. While this is likely because generative problems solve a more general problem before modeling the posterior probabilities, they want to demonstrate when this belief is mistaken. On the other hand, generative classifiers are advantageous because they have documented better performance when training set sizes are small. Also, the EM methods used in generative classifiers can be more easily extended to handle missing data. They conclude that discriminative methods have a lower asymptotic error, while generative methods approach their higher asymptotic error much faster.

While we did not systematically repeat the results of [1] using a Support Vector Classifier, in order to allow for a direct comparison with our Multi-Class sLDA model, our classification results lend preliminary evidence that seems to confirm the widely held belief that discriminative methods are better than generative methods based on test set accuracy. We recommend further experimentation with a focus on varying training set size that directly compares Support Vector methods and Multi-Class sLDA in order to see how they relate to the conclusion of [7]. In summary, as the training set

size increases, we should expect to see Multi-Class sLDA to initially do better, but for the Support Vector Classifier to eventually catch up, and outperform, the accuracy of Multi-Class SLDA.

The authors of [8], directly extended the conclusions drawn from [7] and constructed a hybrid generative/discriminative model. The hybrid model is partly generative (Naive Bayes) and partly discriminative (Logistic Regression). In their model, a large subset of the parameters are trained to maximize the generative component (joint probability, $P(y, x)$) and a much smaller subset of the parameters are trained to maximize the discriminative component (conditional probability, $P(y|x)$). This method allows for control over groupings of variables based on their contribution to the classification decision. Also, this method allows for improvement of accuracy and coverage of the Naive Bayes' model, which makes independence assumptions not reflected by the observable data. In their work, they find that the number of examples needed to fit the discriminative parameters increases only as the logarithm of vocabulary size and document length. The hybrid model to get significantly more accurate classification results and class posterior probabilities that are more representative of the empirical error rates.

Using the results of [8] as an inspiration, a proposed extension to the work here would be to build a hybrid model for this classification task for our problem. However, before we can move in that direction, we need to evaluate whether Multi-Class sLDA, a Topic Model, is amenable to extensions that were applied to the Naive Bayes Classifier. We believe that a hybrid model can be used to better incorporate domain knowledge in our problem, which maybe currently hindering our predictive performance.

Acknowledgments

Thanks to Professor Noah A. Smith for advising and Assistant Dean Mark Stehlik for managing the Senior Thesis program. Thanks to Kogan et. al. [1] for collecting, preprocessing and publishing the Financial Reports and Volatility Measurements used in this work. Thanks to Blei et. al. [2] and Wang et. al. [4] for releasing the source code for their LDA and Multi-Class SLDA implementations, respectively. Final thanks to Chong Wang of [4] for his feedback in helping resolve issues with the Multi-Class SLDA implementation, which was critical to the completion of this work.

References

[1] S. Kogan, D. Levin, B. Routledge, J. Sagi, and N. Smith. Predicting risk from financial reports with regression. In *Proc. NAACL Human Language Technologies Conf.*, 2009.

[2] D. M. Blei, A. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *JMLR*, 3:9931002, 2003

[3] D. M. Blei and J. D. McAuliffe. Supervised Topic Models. In *NIPS*, 2007.

[4] C. Wang, D. M. Blei, and L. Fei-Fei. Simultaneous image classification and annotation. In *CVPR*, 2009.

[5] J. Friedman, T. Hastie and R. Tibshirani. Regularized Paths for Generalized Linear Models via Coordinate Descent. April, 2009.

[6] D. Jurafsky and J. Martin. *Speech and language processing*. Prentice Hall, 2000.

[7] A. Y. Ng, and M. I. Jordan. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In *NIPS* 14, 2001.

[8] R. Raina, Y. Shen, A. Ng and A. McCallum. Classification with hybrid generative/discriminative models. In *NIPS* 16, 2004.

[9] F. Fama. 1970. Efficient capital markets: A review of theory and empirical work. *Journal of Finance*, 25(2):383417.

[10] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2009. Print

[11] H. Zou, T. Hastie. Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society. Series B 67, 301-320, 2005.