Machine Learning of Mathematical Documents

Longphi Nguyen*, Laila Rizvi, Ying Shi*, Jesús De Loera, Mark Junod

April 12, 2013

*Funded by NSF

Contents

1 Introduction

Text classification is a widely explored problem in machine learning especially with the advancement of technology. Typically, in text classification, many documents are grouped into two categories. To get a machine to automatically group the documents, a common approach is to use a training set. Based on certain features, such as document length and content, a computer can be used to automatically categorize the documents with good accuracy. For example, creating a spam filter to classify "spam" or "ham" emails is explained in [13]. Moreover, applying machine learning to find patterns in news articles has been of particular interest (see [12]). In such cases, words that are deemed relevant to a topic or are heavily covered in the news can be used to summarize the important events at a particular time.

This paper explores the idea of text summarization, but rather than being applied to the news, we apply machine learning to mathematical abstracts. The idea is to summarize a certain professor's published papers, which may be of particular interest to, say, students looking for a mentor. To do this, we use ideas shown in [4] to categorize the abstracts as relevant or nonrelevant to a professor's works. That is, we find phrases known as *features* to identify a professor's abstracts. The feature selection methods used are known as *phrase co-occurrence, correlation screening, L1-regularized logistic regression* (L1LR), and *L1-regularized linear regression* (LASSO).

The dataset consists of 1500 abstracts for 40 professors who work on various topics such as topology, partial differential equations, statistics, and so on. We will show results for Jesús De Loera, who has graciously verified our summarizers.

2 Modeling and Solving the Problem

We start by doing some basic preprocessing our data. The texts are converted to lowercase, punctuation is removed, and numbers are removed. Additionally, certain low-information words such as "for" and "the" are ignored. The ignored words are known as stopwords.

After preprocessing the data, we now form X. We let $D := \{1, 2, \ldots, n\}$ be the set of documents indexed by the integers, and let P be the set of one-word and two-word phrases in our dataset. Then, we have $X \in \mathbb{R}^{n \times p}$, where $n = |D|$ and $p := |P|$ with $|\cdot|$ being the cardinalities. The elements of X are denoted by x_{ij} which is the number of times phrase $j \in P$ appears in document $i \in D$. The columns are said to be the *features* and the rows are the *documents*. Hence, X is a features-document matrix, which we simply call our features matrix.

However, it may be preferrable to rescale X to reduce the probability that our feature selection methods will select an insignificant term. The two rescaling approaches that will be compared are known as the L^2 and the term frequency-inverse document frequency (*tf-idf*) rescaling techniques, as seen in [4]. These methods help reduce the variance and weight of high-frequency features (see [5]).

Let $X^{(l)} = [x_{ij}^{(l)}] \in \mathbb{R}^{n \times p}$ be a L^2 rescaled version of X when its columns are normalized under the L^2 norm:

$$x_{ij}^{(l)} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}}. \tag{1}$$

Let $X^{(t)} = [x_{ij}^{(t)}] \in \mathbb{R}^{n \times p}$ be a *tf-idf* rescaled version of X when it is rescaled as follows:

$$x_{ij}^{(t)} := \frac{x_{ij}}{q_i} \log(\frac{n}{d_j}), \text{ where } q_i = \sum_{j=1}^{p} x_{ij} \text{ and } d_j = \sum_{i=1}^{n} \mathbf{1}\{x_{ij} > 0\}, \tag{2}$$

where $\mathbf{1}\{x_{ij} > 0\}$ is an indicator function equal to 1 when $x_{ij} > 0$ and equal to 0 otherwise.

Now, we form a vector $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ for the classification. To do this, we define a subject of interest known as the *query*. In our case, we are interested in certain professors, so our query is a professor's name. If document $i = 1, 2, \ldots, n$ is written by the professor of interest, then $y_i \overset{set}{=} 1$, otherwise $y_i \overset{set}{=} -1$. For simplicity, we denote the set of documents related to the query (i.e. written by the professor of intersest) as D_+, and the set of documents not related to the query (i.e. not written by the professor of intersest) as D_-.

3 Feature Selection Methods

We now discuss the feature selection methods for obtaining our summarizers. The four feature selecition methods we use are *co-occurrence, correlation, L1-penalized linear regression (LASSO),* and *L1-penalized logistic regression (L1LR)*. Using four different methods will reveal some important words that a single method on its own would not have found. We note that we discuss the four methods using the unweighted matrix X, however, they can also be applied to $X^{(l)}$ and $X^{(t)}$.

Co-Occurence is a feature selection method which selects the words that appear the most often or have the greatest weight in the positively marked text. For each phrase $j \in P$, compute the relevance score s_j as follows:

$$s_j = E(x_j | y = 1) = \frac{1}{|D_+|} \sum_{i \in D_+} x_{ij} \tag{3}$$

where $x_j = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T$ and $E(x_j | y = 1)$ is the expectation of x_j given $y = 1$. We then sort the s_i as $s'_1 \geq s'_2 \geq ... \geq s'_p$ and select the first $1 \leq k \leq p$ terms as our summarizer. That is, the phrases associated with $s'_1, s'_2, \ldots, s'_k$ form our summarizer.

Pearson's correlation is a measure of the linear dependence between two or more variables. Geometrically, correlation is the cosine between two vectors as explained in [17]. Specifically, it is the cosine of the variance vectors of two random variables. As such, correlation coefficients range from -1 to +1, with the extremes being a perfect negative and perfect positive correlation respectively. Variables with a perfectly positive or negative correlation lie on the same line, whereas variables with a correlation of zero have no linear relationship. Correlation screening calculates the correlation between a feature column $\boldsymbol{x}_j$ and $\boldsymbol{y}$, and selects features with the largest absolute correlation. For each phrase $j \in P$, compute the relevance score s_j as follows:

$$s_j = |corr(x_j, y)| = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4}$$

where $\bar{x}$ is the sample mean of x and $\bar{y}$ is the sample mean of y.

Next, we sort the s_i as $s'_1 \geq s'_2 \geq ... \geq s'_p$ and select the first $1 \leq k \leq p$ terms as our summarizer. This method selects phrases that tend to appear in the positively marked text and not in the negatively marked text because they have a relatively strong linear relationship with the subject.

While co-occurence and correlation are quite similar, they are not identical. Co-occurrence selects phrases because they co-occur frequently, on average, together in the positively marked text. However, the problem is that these words might not always be related to the query. The authors of these works might simply use these phases often, causing their co-occurence scores to increase. On the other hand, correlation screening selects phrases that have a have a higher chance of occurring with the query of interest because they are actually relevant to the subject. For instance, an author might write a paper on geometry and use fruits as an example of geometric shapes in one of his documents. "Fruits" would then have a high co-occurrence score, but would have a low correlation score because it is not relevant to the author's other documents.

L1-penalized Linear Regression (LASSO) is a widely-used feature selection model which is based off of Least Square Linear Regression (LSLR). LSLR is a well-known method used to find a linear relationship between an explanatory variable and a response variable (see [3]). In our case, each y_i in the $\mathbf{y}$ vector is a response variable, and the entries in the corresponding i^{th} row of the X matrix are the p explanatory variables. That is, the $\mathbf{y}$ vector is the response vector and the X matrix is the explanatory matrix. Assuming a linear relationship between $\mathbf{y}$ and X, we write $\mathbf{y}$ as a linear combination of the p columns of X plus some error terms:

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\gamma}.$$

The coefficient vector $\boldsymbol{\beta} := [\beta_1, \beta_2, \ldots, \beta_p]^T$ is the parameter of interest. Thus, by linear relationship, we mean that $\mathbf{y}$ is linear in $\boldsymbol{\beta}$ (see [1]). The intercept vector $\boldsymbol{\gamma}$ is constant. The way to estimate $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ is through minimizing the sum of square of the error terms:

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\boldsymbol{y} - X\boldsymbol{\beta} - \boldsymbol{\gamma}||^2.$$

A naive approach for choosing a small number of features is by to select the $\hat{\beta}_i > \epsilon$ for a selected $\epsilon > 0$. The corresponding features can be selected as the summarizer. However, this result from LSLR is difficult to interpret, because an excessive number of coefficients will potentially end up statistically significant. Since our goal is to find a sparse set of features to summarize the documents, we could simply choose the first few features corresponding to the most significant coefficients from the LSLR result. Yet, those features are not representative empirically. In theory, the reason is that the result from LSLR is largely influenced by the outliers since all of the p phrases are included in the linear combination. To minimize the effect of outliers, we penalize them by adding a L_1-norm penalty term: $\lambda \sum_{j=1}^{p} |\beta_j|$. The resulting objective function is called the LASSO:

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} ||\boldsymbol{y} - C\boldsymbol{\beta} - \boldsymbol{\gamma}||^2 + \lambda \sum_{j=1}^{p} |\beta_j|, \text{ where } \lambda > 0. \tag{5}$$

Here, λ is a penalty parameter which can force $\hat{\boldsymbol{\beta}}$ to be sparse. The higher the λ, the more sparse $\hat{\boldsymbol{\beta}}$ is. Implicit in (5) is that there is a trade-off between the sum of square of the errors and the nonzero coefficients in $\hat{\boldsymbol{\beta}}$. If a coefficient's existence cause the sum of square of the errors to be large, it will be regularized to zero. Due to this trade-off, most of the coefficients will turn out to be zero as desired [3]. Note that the intercept γ is not penalized. If it is, the intercept will get close to zero, which indicates that the number of times when y_i takes on 1 equals to the number of times when y_i takes on -1. An easy way to see this is to consider a special case where the C matrix is a zero matrix. Then, the intercept is the average of the y_i's. This average being close to zero indicates that y_i takes on 1 or -1 for approximately an equal number of times. In our case, this is not true since y_i's typically equal -1, so we do not penalize $\boldsymbol{\gamma}$ (see [4]).

L1-penalized Logistic Regression (L1LR) is another type of regression analysis, but is specifically designed for classification in which the response variable is categorical. Our case is a binary logistic regression since each entry of the response vector $\mathbf{y}$ only takes on 1 or -1. The parameter of interest is the probability of y_i being 1, written as $P(y_i = 1)$, in each sample (see [9]). We would first try to directly write $P(y_i = 1)$ as a linear combination of the features represented by the i^{th} row of X:

$$P(y_i = 1) = \boldsymbol{x_i}^T\boldsymbol{\beta} + \gamma, \tag{6}$$

where $\boldsymbol{x_i} = \begin{pmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_p \end{pmatrix}$, and γ is a constant (see [9]).
One problem for this model is that the right hand side can take values greater than 1 or less than 0, while the probability on the left hand side cannot. A remedy is to take the natural logarithm of the probability ratio:

$$\ln \frac{P(y_i = 1)}{P(y_i = -1)}.$$

This is called $logit(P(y_i = 1))$, which can match the range of the right hand side of (6). Moreover, $logit(P(y_i = 1))$ is symmetric, i.e. $logit(P(y_i = 1)) = -logit(P(y_i = -1))$ as explained in [9].
Now we can model $logit(P(y_i = 1))$ as a linear combination of the features:

$$\ln \frac{P(y_i = 1)}{P(y_i = -1)} = \boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma.$$

Since it is more intuitive to think in terms of probabilities, we solve for $P(y_i = 1)$ by substituting $P(y_i = -1)$ with $1 - P(y_i = 1)$:

$$\ln \frac{P(y_i = 1)}{1 - P(y_i = 1)} = \boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma.$$

$$\frac{P(y_i = 1)}{1 - P(y_i = 1)} = \exp(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma).$$

$$P(y_i = 1) = \frac{1}{1 + \exp(-(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma))}.$$

This is called the *logistic model*. Similarly, we can also solve for $P(y_i = -1)$:

$$P(y_i = -1) = \frac{\exp(-(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma))}{1 + \exp(-(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma)}.$$

The general form of the probability for y_i is

$$P(y_i) = \frac{\exp(y_i(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma))}{1 + \exp(y_i(\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma))}. \tag{7}$$

If $\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma = 0$, $P(y_i)$ is 0.5, which means that it is equally possible for y_i to take 1 or -1 . If $\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma = 1$, $P(y_i = 1)$ and $P(y_i = -1)$ are 0.73 and 0.27 respectively. If $\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma = -1$, the reverse holds. When $\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma > 1$, $P(y_i = 1)$ quickly converges to 1. When $\boldsymbol{x_i}^T \boldsymbol{\beta} + \gamma < -1$, $P(y_i = 1)$ quickly converges to 0. The region in between is ambiguous, meaning that y_i is not strongly likely to take one side [2].
In order to estimate the parameter $\boldsymbol{\beta}$ from this logistic model, we will use the technique of maximizing the log-likelihood. Likelihood is the probability that a particular sample can occur given different values of the parameter in the model (see [6]). In our case, the sample is the set of documents D, and the parameter is $\boldsymbol{\beta}$. Since D is given, the parameter $\boldsymbol{\beta}$ that can maximize the likelihood of D can be found. For simplicity, each document in D is assumed to be independent, meaning that the occurrence of one document does not affect the probability of the others (see [6]). Then the likelihood function for D is

$$\prod_{i=1}^{n} P(y_i).$$

By taking the logarithm, we obtain the so-called log-likelihood function:

$$\begin{aligned}\log \prod_{i=1}^{n} P(y_i) &= \sum_{i=1}^{n} \log P(y_i) \\ &= \sum_{i=1}^{n} \log \frac{\exp(y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))}{1+\exp(y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))} \\ &= \sum_{i=1}^{n} \log \frac{1}{1+\exp(-y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))} \\ &= -\sum_{i=1}^{n} \log(1+\exp(-y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))).\end{aligned}$$

We maximize the log-likelihood function or, equivalently, minimize the negative of it:

$$\arg\min_{\beta,\gamma} \sum_{i=1}^{n} \log(1+\exp(-y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))).$$

In order to force $\boldsymbol{\beta}$ to be sparse, the L_1-norm penalty term is added (see [4]). Thus, the complete objective function for L1LR is

$$(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\gamma}}) = \arg\min_{\boldsymbol{\beta},\boldsymbol{\gamma}} \sum_{i=1}^{n} \log(1+\exp(-y_i(\boldsymbol{x_i}^T\boldsymbol{\beta}+\gamma))) + \lambda \sum_{j=1}^{p} |\beta_j|. \tag{8}$$

Although the original purpose of L1LR is to classify the new samples based on the features selected, we are only interested in the selected features themselves. Though these features can be used for classification, we will only consider them as summarizers. As shown above, L1LR, in theory, may give better results because of the binary nature of our experiment.

3.1 Convex Optimization

We now explain how the *L1LR* and *LASSO* objective functions can be solved by noting that they are convex. A set is convex if the line segment connecting two arbitrary points of the set completely lies inside the set. Examples of a convex set in $\mathbb{R}^2$ include squares, circles, ellipsoids and so on [16]. Mathematically, if a set $V \subseteq \mathbb{R}^n$ is convex, for any $x, y \in V$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x + (1-\theta)y \in V.$$

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of f, written as $D(f)$, is a convex set and if for all $x, y \in D(f)$ and $0 \leq \theta \leq 1$

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y).$$

Function $-f$ is then concave [14]. Convex sets and a convex function $f : D(f) \to \mathbb{R}$ can be connected through the epigraph of the function f, which is defined as

$$epi(f) = \{(x, y) | x \in D(f), f(x) \leq y\}.$$

This can be used to show that a function is convex, according to the following theorem.

Theorem 1. *A function $f : D(f) \to \mathbb{R}$ is a convex function if and only if* epi(f) *is a convex set.*

The penalty term in the LASSO and L1LR objective functions is a sum of the absolute functions, which can be written as $f(z) = |z|$. Since $epi(f(z))$ is a convex set, $f(z)$ is a convex function. The penalty term, which sums the absolute functions, is also convex due to the following theorem.

Theorem 2. *If $f, g : D(f) \to \mathbb{R}$ are convex functions, then the sum of f and g is also a convex function.*

Next theorem is useful to prove that the complete LASSO and L1LR objective functions are convex.

Theorem 3. *Let $f : D(f) \to \mathbb{R}$ have a continuous second derivative. f is convex on the convex set $D(f)$ if and only if $f''(x) \geq 0$ for all $x \in D(f)$.*

The LASSO objective function without a penalty term is

$$\arg\min_{\beta,\gamma} ||\mathbf{y} - X\boldsymbol{\beta} - \boldsymbol{\gamma}||^2, \tag{9}$$

which is convex. Equation (9) can be viewed as a sum of functions in the form of $f(z) = z^2$. Since $f(z)'' = 2$, (9) is convex by Theorem 2 and 3.

The L1LR objective function without a penalty term can also be proved to be convex. Let us rewrite the log-likelihood function as

$$\begin{aligned}\log \prod_{i=1}^{n} P(y_i) &= -\sum_{i=1}^{n} \log(1 + \exp(-y_i(\boldsymbol{x_i}^T\boldsymbol{\beta} + \gamma))) \\ &= -\sum_{i=1}^{n} f(\boldsymbol{x_i}^T\boldsymbol{\beta} + \gamma),\end{aligned}$$

where f is the *logistic loss function* in a form of

$$f(z) = 1 + \exp(-z).$$

Then,

$$\begin{aligned}f(z)' &= -\frac{\exp(-z)}{1 + \exp(-z)} \\ f(z)'' &= \frac{\exp(-z)}{(1 + \exp(-z))^2}.\end{aligned}$$

Since $f(z)'' > 0$, the logistic loss function is convex by Theorem 1. The log-likelihood function is then concave. Thus, we take the negative of it to make it convex so that we can use convex optimization methods to solve it [14]. If we add the penalty term, the complete LASSO and L1LR objective functions are proved to be convex.

Convex optimization The standard form of a general optimization problem is

$$\begin{aligned}&\text{minimize } f_0(x) \\ &\text{subject to } f_i(x) \geq 0, i = 1, 2, ..., m.\end{aligned}$$

The optimal solution is the $x \in \mathbb{R}^n$ that can minimize the objective function $f_0(x)$ with constraints. Note that the constraints can also be strictly equal.

The convex optimization problem requires that the objective function and the inequality constraint functions are convex, and that the equality constraint functions are linear. An important feature described

in the following theorem distinguishes the convex optimization problems from the general optimization ones [14].

Theorem 4. *If $f_0 : D(f) \to \mathbb{R}$ is a convex function, then every local optimum is a global optimum.*

Consequently the global minimum of the LASSO and L1LR objective functions are guaranteed. Also, note that our objective functions are unconstrained. One method to solve an unconstrained convex optimization problem is Newton's method [14]. According to Newton's method, the optimal solution for our case is the $\boldsymbol{\beta}$ such that

$$\nabla f(\boldsymbol{\beta}) = 0, \text{ where } \nabla f(\boldsymbol{\beta}) = (\frac{\partial}{\partial \beta_1} f(\boldsymbol{\beta}), ..., \frac{\partial}{\partial \beta_p} f(\boldsymbol{\beta}))^T$$

Here $f(\cdot)$ refers to the LASSO and L1LR objective functions without the penalty term, because the Newton's method requires differentiability. Since the penalty term makes the objective functions not differentiable. To solve the objectives including the penalty terms, an efficient alternative can be to use the cyclical coordinate descent method which can be seen in [15].

4 Methods for Reducing the Dimensions of X

Before showing our results, we would like to note that it is possible to greatly reduce the dimensions of our features matrix and still retain a fairly good summarizer. The significance of this is that the computational time for the feature selection methods is greatly reduced, as discussed in Section 6. We first introduce Principal Component Analysis (PCA) and its sparse version known as Sparse PCA (SPCA). Next we introduce a new method developed by De Loera and Junod (2013) that creates corpus-based stop word lists. The words and bigrams in these stop word lists are then removed from our feature matrix, as described in Section 2.

4.1 Principal Component Analysis (PCA)

Principal Component Analysis is a method often employed to reduce a high dimensional data matrix into a lower dimensional one, while also minimizing the amount of lost information. For a $n \times p$ matrix, PCA seeks to find p uncorrelated linear combinations of the p variables. These linear combinations are known as Principal Components (PCs), and they represent the directions with maximum variability in the data. The PCs are found in the following manner, as shown in [9].

Principal Components. Let $\boldsymbol{x}^T = [x_1, x_2, \ldots, x_p]$ be a random vector centered about 0. The covariance matrix of $\boldsymbol{x}$ is $\Sigma = \boldsymbol{x}\boldsymbol{x}^T$. Let the eigenvalue-eigenvector pairs of Σ be $(\lambda_1, \boldsymbol{e}_1), (\lambda_2, \boldsymbol{e}_2), \ldots, (\lambda_p, \boldsymbol{e}_p)$ with $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p$. The i^{th} principal component is given by the random vector

$$y_i = \boldsymbol{e}_i^T \boldsymbol{x} = e_{i1}x_1 + e_{i2}x_2 + \ldots + e_{ip}x_p,$$

with the property that

$$Cov(y_i, y_k) = \boldsymbol{e}_i^T \Sigma \boldsymbol{e}_k = \begin{cases} \lambda_i, & i = k \\ 0, & i \neq k \end{cases} \text{ for } i, k = 1, 2, \ldots, p.$$

That is, the PCs are uncorrelated and their variances are equal to the eigenvalues of Σ. Using all of the PCs can explain the variance in the data. However, some PCs explain relatively little variance, so they are

often ignored. Then, we can select $k(\leq p)$ PCs without losing much information. Often, a scree-plot of the eigenvalues is used to determine what k to use. The scree-plot puts (i, λ_i) on a graph, and k is determined such that λ_{k+1} is relatively "small." What exactly is meant by small depends on the application.

The PCA problem can be transformed to regression-type problem, as shown in [8] and [1]. We omit the proof of the following theorem.

Theorem 5. *Let $\alpha, \beta \in \mathbb{R}^{p\times k}$, $\boldsymbol{\beta}_j$ be the j^{th} column vector of β, $\boldsymbol{x}_i$ be the i^{th} row vector of the matrix $X \in \mathbb{R}^{n\times p}$, and $k \leq p$. Then for some $\gamma > 0$, the first k principal components can be found by*

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \alpha\beta^T \boldsymbol{x}_i\|_2^2 + \gamma \sum_{j=1}^{k} \|\boldsymbol{\beta}_j\|_2^2, \text{ subject to } \alpha^T\alpha = I_k.$$

Letting $\hat{\beta} =: [\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, \ldots, \hat{\boldsymbol{\beta}}_k]$ and $\hat{\boldsymbol{v}}_i := \frac{\hat{\boldsymbol{\beta}}_i}{\|\hat{\boldsymbol{\beta}}_i\|_2}$, then the i^{th} PC is $y_i = X\hat{\boldsymbol{v}}_i$.

Sparse Principal Component Analysis (SPCA), on the other hand, attempts to make the loadings $\boldsymbol{v}_i$ in Theorem (5) sparse so that the principal components are a linear combination of only some variables. The SPCA problem can be formulated in many ways, but we will discuss it in a regression framework as described in [8]. Recall that Theorem (5) shows how PCA can be written as a regression-type problem. To encourage sparsity in the loadings, then, we can use the L_1-norm penalty as we have for $L1LR$ and $LASSO$. Adding a L_1-norm penalty gives us the following problem.

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \alpha\beta^T \boldsymbol{x}_i\|_2^2 + \gamma \sum_{j=1}^{k} \|\boldsymbol{\beta}_j\|_2^2 + \sum_{j=1}^{k} \lambda_j \|\boldsymbol{\beta}_j\|_1, \text{ subject to } \alpha^T\alpha = I_k. \tag{10}$$

Note that the penalty terms λ_j may differ so that the loadings of different PCs are penalized based on the PC's significance.

And similar to before, we let $\hat{\beta} =: [\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2, \ldots, \hat{\boldsymbol{\beta}}_k]$ and $\hat{\boldsymbol{v}}_i := \frac{\hat{\boldsymbol{\beta}}_i}{\|\hat{\boldsymbol{\beta}}_i\|_2}$. Then the i^{th} sparse PC is $y_i = X\hat{\boldsymbol{v}}_i$.

4.2 Corpus-Based Stop Word Lists

In Section 2 we briefly discussed stop words as low information words we can remove from the features matrix without negatively impacting our selection of sparse summarizers. Typical stop words such as "the" or "and" are necessary grammatically, but typically add very little information. For example, most internet search engines have developed lists of several hundred typical stop words that are automatically removed from search queries prior to running the search algorithm. The reason is words like "the" show up far too often to help determine if a particular webpage relates to the query or not.

Unfortunately, in subject specific corpora like ours, these pre-made stop word lists often contain words that are rarely, if ever, used, such as the word "anybody". At the same time, there are many words that in a mathematical context can be thought of as stop words, such as "function", even though the same words would be very informative in a corpus consisting of a random selection of English documents. Below we describe a method that deals with these issues by creating a corpus-specific stop word list, allowing us to eliminate thousands of terms from our features matrix. See [**?**] for full details of the method.

Building a Semantic Graph Assume we have a corpus and a *corpus-based semantic measure* between words in the corpus. A semantic measure is any function $m : P \times P \to \mathbb{R}_{\geq 0}$ that is created based only on the given corpus. More concretely, a semantic measure compares two words and returns a nonnegative number that describes how similar the two words are semantically. Not all words need to be directly comparable. For examples of corpus-based semantic measures, see [**?**, **?**, **?**].

Once we have a semantic measure, we can create a semantic graph of the entire corpus. Call this graph G_P. The words of the corpus form the nodes of the graph, and two words w_i and w_j are connected if $m(w_i, w_j)$ exists. We then weight the edge with an appropriate transformation of the semantic measure. The reason we take a transformation is we want higher weights to correspond to words that are less similar, and unfortunately most of the corpus-based semantic measures behave in the opposite way. That is, higher values correspond to words that are semantically close together.

So we now have a graph G_P of the semantic connections between the words in the corpus. High weights on edges mean the words are not similar, while low weights mean the words are similar. At this point we can create our stop word list.

Creating the Stop Word List Building the stop word list takes two steps. First we create a small list of representative stop words. In the second step we use our representative list to find the rest of the stop words in the corpus. From a machine learning point of view, the representative list is our training data we use to classify the rest of the stop words.

5 Results

Here, we show our summarizers when using "Jesús De Loera" as the query. For three different weighting schemes, we list our summarizers in two tables. These tables are detailed as follows.

- Table 1: The four columns correspond to the models used: Coocurrence, Correlation, L1LR, and LASSO. Each column contains a list of words that the corresponding model selected, ordered from top to bottom by what the model selects as most to least significant. Convergence in L1LR and LASSO was decided when at most 15 features were selected. Let us denote the number of terms selected by L1LR and LASSO by n_1 and n_2 respectively. The number of terms kept by Coocurrence and Correlation was $min(10, max(n1, n2))$. Calculations were done using the full matrix which has dimensions 1558×124523..

- Table 2: The table is similar to Table 1, but the calculations were done using the reduced matrix obtained after SPCA. The dimensions for the reduced matrix is 1558×29043. As shown in the following tables, the results from the full matrix and the reduced matrix differ little, and often only differ by the order of the words.

For reference, we list a description of the matrices that were used in our calculations. For $y \in \{1, 2\}$, we have the following descriptions:

X_1 is the unweighted features matrix with its columns being the unigrams and bigrams of our data.

X_2 is X_1 after our matrix dimension methods have been applied.

$X_y^{(l)}$ is the l^2 weighted version of X_y.

$X_y^{(t)}$ is the tf-idf weighted version of X_y.

X_{b1} is the unweighted features matrix with its columns being the bigrams of our data.

X_{b2} is X_{b1} after our matrix dimension methods have been applied.

$X_{by}^{(t)}$ is the tf-idf weighted version of X_{by}.

$X_{by}^{(t)}$ is the tf-idf weighted version of X_{by}.

$X_{by}^{(l)}$ is the l^2 weighted version of X_{by}.

$X_{by}^{(t)}$ is the tf-idf weighted version of X_{by}.

5.1 Results From Features Matrix of Unigrams and Bigrams Without Weighting

	cooc	corr	l1lr	lasso
1	with	barvinok	related system	related system
2	this	polynomial time	polynomial time	between regular
3	polynomial	report	dimensional simplices	polynomial time
4	on	between regular	report	barvinok
5	be	regular triangulations	between regular	report
6	by	transportation	transportation	fully polynomial
7	polytopes	point configuration	barvinok	transportation
8	integer	knapsack problems	fully polynomial	dimensional simplices
9	authors	related system	triangulations	triangulations
10	paper	nullstellensatz certificates	regular triangulations	

Table 1: Summarizers when the query is "Jesús De Loera" and X_1 is used.

	cooc	corr	l1lr	lasso
1	with	barvinok	related system	related system
2	this	polynomial time	polynomial time	between regular
3	polynomial	report	dimensional simplices	polynomial time
4	on	between regular	report	barvinok
5	be	regular triangulations	between regular	report
6	by	transportation	transportation	fully polynomial
7	polytopes	point configuration	barvinok	transportation
8	integer	knapsack problems	fully polynomial	dimensional simplices
9	authors	related system	triangulations	triangulations
10	paper	nullstellensatz certificates	regular triangulations	

Table 2: Summarizers when the query is "Jesús De Loera" and X_2 is used.

5.2 Results From Features Matrix of Unigrams and Bigrams With L^2 Weighting

	cooc	corr	l1lr	lasso
1	barvinok	barvinok	related system	related system
2	polynomial time	polynomial time	polynomial time	between regular
3	report	report	dimensional simplices	polynomial time
4	regular triangulations	between regular	report	barvinok
5	between regular	regular triangulations	between regular	report
6	transportation	transportation	transportation	fully polynomial
7	integer	point configuration	barvinok	transportation
8	polytope	knapsack problems	fully polynomial	dimensional simplices
9	point configuration	related system	triangulations	triangulations
10	knapsack problems	nullstellensatz certificates	regular triangulations	

Table 3: Summarizers when the query is "Jesús De Loera" and $X_1^{(l)}$ is used.

	cooc	corr	l1lr	lasso
1	barvinok	barvinok	related system	related system
2	polynomial time	polynomial time	polynomial time	between regular
3	report	report	dimensional simplices	polynomial time
4	regular triangulations	between regular	report	barvinok
5	between regular	regular triangulations	between regular	report
6	transportation	transportation	transportation	fully polynomial
7	integer	point configuration	barvinok	transportation
8	polytope	knapsack problems	fully polynomial	dimensional simplices
9	point configuration	related system	triangulations	triangulations
10	knapsack problems	nullstellensatz certificates	regular triangulations	

Table 4: Summarizers when the query is "Jesús De Loera" and $X_2^{(l)}$ is used.

5.3 Results From Features Matrix of Unigrams and Bigrams With tf-idf Weighting

	cooc	corr	l1lr	lasso
1	with	barvinok	related system	related system
2	this	polynomial time	polynomial time	between regular
3	polynomial	between regular	dimensional simplices	polynomial time
4	integer	transportation	report	barvinok
5	polytopes	triangulations	between regular	report
6	authors	nullstellensatz	transportation	fully polynomial
7	by	report	barvinok	transportation
8	triangulations	boston boston	fully polynomial	dimensional simplices
9	problems	rational functions	triangulations	triangulations
10	be	instances	regular triangulations	

Table 5: Summarizers when the query is "Jesús De Loera" and $X_1^{(t)}$ is used.

	cooc	corr	l1lr	lasso
1	with	barvinok	related system	related system
2	integer	polynomial time	polynomial time	between regular
3	this	transportation	dimensional simplices	polynomial time
4	polytopes	between regular	report	barvinok
5	polynomial	triangulations	between regular	report
6	authors	nullstellensatz	transportation	fully polynomial
7	triangulations	boston	barvinok	transportation
8	problems	fixed dimension	fully polynomial	dimensional simplices
9	be	report	triangulations	triangulations
10	by	rational functions	regular triangulations	

Table 6: Summarizers when the query is "Jesús De Loera" and $X_2^{(t)}$ is used.

5.4 Results From Features Matrix of Bigrams Without Weighting

The following tables are obtained when we only use bigrams in our list of phrases. We let X_{b1} be the full matrix as before, but without the single-word columns. Similarly, we let X_{b2} be the reduced matrix as before, but without the single-word columns.

	cooc	corr	l1lr	lasso
1	can be	polynomial time	related system	between regular
2	polynomial time	between regular	between regular	related system
3	no mr	regular triangulations	polynomial time	polynomial time
4	gel fand	point configuration	integer solutions	integer solutions
5	based on	knapsack problems	regular triangulations	regular triangulations
6	this paper	related system	knapsack problems	knapsack problems
7	grbner bases	nullstellensatz certificates	way transportation	way transportation
8	fand tsetlin	fully polynomial		
9	at least	time approximation		
10	integer programming	approximation scheme		

Table 7: Summarizers when the query is "Jesús De Loera" and X_{b1} is used.

	cooc	corr	l1lr	lasso
1	can be	polynomial time	related system	between regular
2	polynomial time	between regular	between regular	related system
3	no mr	regular triangulations	polynomial time	polynomial time
4	gel fand	point configuration	integer solutions	integer solutions
5	based on	knapsack problems	regular triangulations	regular triangulations
6	this paper	related system	knapsack problems	knapsack problems
7	grbner bases	nullstellensatz certificates	way transportation	way transportation
8	fand tsetlin	fully polynomial		
9	at least	time approximation		
10	integer programming	approximation scheme		

Table 8: Summarizers when the query is "Jesús De Loera" and X_{b2} is used.

5.5 Results From Features Matrix of Bigrams With L^2 Weighting

The following tables are obtained when we only use bigrams in our list of phrases. We let X_{b1} be the full matrix as before, but without the single-word columns. Similarly, we let X_{b2} be the reduced matrix as before, but without the single-word columns. We then weight the matrices using L^2 weighting.

	cooc	corr	l1lr	lasso
1	polynomial time	polynomial time	related system	between regular
2	regular triangulations	between regular	between regular	related system
3	between regular	regular triangulations	polynomial time	polynomial time
4	point configuration	point configuration	integer solutions	integer solutions
5	knapsack problems	knapsack problems	regular triangulations	regular triangulations
6	related system	related system	knapsack problems	knapsack problems
7	nullstellensatz certificates	nullstellensatz certificates	way transportation	way transportation
8	ehrhart series	ehrhart series		
9	fully polynomial	fully polynomial		
10	time approximation	time approximation		

Table 9: Summarizers when the query is "Jesús De Loera" and $X_{b1}^{(l)}$ is used.

	cooc	corr	l1lr	lasso
1	polynomial time	polynomial time	related system	between regular
2	regular triangulations	between regular	between regular	related system
3	between regular	regular triangulations	polynomial time	polynomial time
4	point configuration	point configuration	integer solutions	integer solutions
5	knapsack problems	knapsack problems	regular triangulations	regular triangulations
6	related system	related system	knapsack problems	knapsack problems
7	nullstellensatz certificates	nullstellensatz certificates	way transportation	way transportation
8	ehrhart series	ehrhart series		
9	fully polynomial	fully polynomial		
10	time approximation	time approximation		

Table 10: Summarizers when the query is "Jesús De Loera" and $X_{b2}^{(l)}$ is used with L^2 weighting.

5.6 Results From Features Matrix of Bigrams With tf-idf Weighting

The following tables are obtained when we only use bigrams in our list of phrases. We let X_{b1} be the full matrix as before, but without the single-word columns. Similarly, we let X_{b2} be the reduced matrix as before, but without the single-word columns. We then weight the matrices using $tf - idf$ weighting.

	cooc	corr	l1lr	lasso
1	polynomial time	polynomial time	related system	between regular
2	no mr	between regular	between regular	related system
3	this paper	rational functions	polynomial time	polynomial time
4	can be	boston boston	integer solutions	integer solutions
5	based on	dimensional simplices	regular triangulations	regular triangulations
6	grbner bases	ma mr	knapsack problems	knapsack problems
7	erratum to	regular triangulations	way transportation	way transportation
8	grbner basis	triangulation correspond		
9	rational functions	way transportation		
10	gromov norm	non regular		

Table 11: Summarizers when the query is "Jesús De Loera" and $X_{b1}^{(t)}$ is used.

	cooc	corr	l1lr	lasso
1	polynomial time	polynomial time	related system	between regular
2	this paper	between regular	between regular	related system
3	no mr	point configuration	polynomial time	polynomial time
4	based on	non regular	integer solutions	integer solutions
5	can be	integer solutions	regular triangulations	regular triangulations
6	gel fand	nullstellensatz certificates	knapsack problems	knapsack problems
7	integer programming	boston boston	way transportation	way transportation
8	grbner bases	triangulation correspond		
9	ehrhart polynomials	knapsack problems		
10	gromov norm	ehrhart series		

Table 12: Summarizers when the query is "Jesús De Loera" and $X_{b2}^{(t)}$ is used.

6 Speed Tests

Computational speed plays a role in deciding which of the four methods is preferable. The computational complexities for the methods, using Big-O notation, are as follows. Let v denote the number of nonzero elements in the features matrix X. Then, we have the following computational complexities as explained in [4]:

Co-occurrence and Correlation: $O(v)$

Lasso and L1LR: $O(\text{iteration limit} \times \text{summarizer size})$

Below, we show tables with the computational times of the four feature selection methods. Each method was run 40 times, and the computational times were recorded in seconds. The following tables show the distribution of the recorded times for each method. The columns, from left to right, correspond to the minimum recorded time, 25% quantile, 50% quantile, 75% quantile, maximum recorded time, and average time.

	0%	25%	50%	75%	100%	Average
Coocurrence	0.15	0.16	0.16	0.17	0.24	0.16
Correlation	0.82	0.83	0.83	0.84	0.85	0.83
L1lR	3.84	3.90	3.93	3.96	4.01	3.93
LASSO	3.98	4.07	4.08	4.09	4.14	4.08

Table 13: Distribution of computational times in seconds for four methods when the query is "Jesús De Loera" and the full features matrix is used.

	0%	25%	50%	75%	100%	Average
Coocurrence	0.03	0.04	0.05	0.05	0.08	0.05
Correlation	0.29	0.30	0.31	0.31	0.33	0.31
L1LR	1.42	1.45	1.48	1.51	1.60	1.48
LASSO	1.45	1.49	1.50	1.53	1.55	1.51

Table 14: Distribution of computational times in seconds for four methods when the query is "Jesús De Loera" and the reduced features matrix is used.

A Proof of Theorem (5)

$$\sum_{i=1}^{n} |\boldsymbol{x}_i - \alpha\beta^T\boldsymbol{x}_i|^2 + \gamma\sum_{j=1}^{k}|\beta_j|^2 = tr(\boldsymbol{x}_i^T\boldsymbol{x}_i) - 2tr(\alpha^T\boldsymbol{x}_i^T\boldsymbol{x}_i\beta) + tr(\beta^T(\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)\beta)$$
$$= tr(\boldsymbol{x}_i^T\boldsymbol{x}_i) + \sum_{j=1}^{k}(\beta_j^T(\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)\beta_j - 2\alpha_j^T\boldsymbol{x}_i^T\boldsymbol{x}_i\beta_j)$$

For a given α, this is minimized at $\beta_j = (\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)^{-1}\boldsymbol{x}_i^T\boldsymbol{x}_i\alpha_j$ for $j = 1, 2, \ldots, k$. Equivalently, it is minimized at $\beta = (\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)^{-1}\boldsymbol{x}_i^T\boldsymbol{x}_i\alpha$.

Then, we find an α that maximizes β_j in order to minimize the summation.

$\hat{\alpha} = \arg\max_\alpha tr(\alpha^T\boldsymbol{x}_i^T\boldsymbol{x}_i(\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)^{-1}\boldsymbol{x}_i^T\boldsymbol{x}_i\alpha)$, subject to $\alpha^T\alpha = I_k$

This has the solution $\hat{\alpha_j} = s_j\boldsymbol{v_j}$, where $|s_j| = 1$ and $\boldsymbol{v_j}$ is an eigenvector of $\boldsymbol{x}_i^T\boldsymbol{x}_i(\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)^{-1}\boldsymbol{x}_i^T\boldsymbol{x}_i$ for $j = 1, 2, \ldots, k$.

Then, $\hat{\boldsymbol{\beta}}_i = (\boldsymbol{x}_i^T\boldsymbol{x}_i+\gamma)^{-1}\boldsymbol{x}_i^T\boldsymbol{x}_i\hat{\alpha_i} \propto \boldsymbol{v}_i$.

References

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

[2] K. Koh, S. Kim, and S. Boyd. "An Interior-Point Method for Large-Scale l_1-Regularized Logistic Regression." *Journal of Machine Learning Research* 8 (2007): 1519-1555. Web. 3 Mar. 2013. <*www.stanford.edu/~boyd/papers/l1_logistic_reg.html*>.

[3] M. Kutner, C. Nachtsheim, et al. *Applied Linear Statistical Models*. New York: McGraw-Hill, 2004. Print.

[4] L. Miratrix, J. Jia, B. Gawalt, B. Yu, and L. El Ghaoui. Summarizing Large-scale, Multiple-document News Data: Sparse Methods and Human Validation. In *JASA.*

[5] B. L. Monroe, M. P. Colaresi, and K. M. Quinn (2008). Fighting Words: Lexical Feature Selection and Evaluation for Identifying the Content of Political Conflict. In *Political Analysis* 16(4).

[6] G. Roussas. *An Introduction to Probability and Statistical Inference*. San Diego: Academic Press, 2003. Print.

[7] Y. Zhang and L. El Ghaoui. (2011) Large-scale Sparse Principal Component Analysis with Application to Text Data. In *NIPS.*

[8] H. Zou, T. Hastie, R. Tibshirani. "Sparse Principal Component Analysis." Journal of Computational and Graphical Statistics 15.2 (2006): 265-286. Web. <*www.stanford.edu/~hastie/Papers/spc_jcgs.pdf*>.

[9] R. Johnson and D. Wichern. *Applied Multivate Statistical Analysis, 6th ed.* New Jersey: Pearson Education, 2007. Print.

[10] H. Shen and J. Huang. "Sparse Principal Component Analysis via Regularized Low Rank Matrix Approximation." Journal of Multivariate Analysis 99 (2008): 1015-1034. Web. <*www.sciencedirect.com/science/article/pii/S0047259X07000887*>.

[11] L. El Ghaoui, G. Li, et al. "Sparse Machine Learning Methods for Understanding Large Text Corpora." Conference on Intelligent Data Understanding, July 2011. Web. <*www.eecs.berkeley.edu/~elghaoui/pubs_cidu2011.html*>.

[12] N. Christiani, E. Hensinger, and I. Flaounas. "Modelling and Predicting News Popularity." Pattern Analysis and Applications (2012). Web. <*https://patterns.enm.bris.ac.uk/people/nello-cristianini*>.

[13] S. Russell and P. Norvig. *Artificial Intelligence, 3rd ed.* New Jersey: Pearson Education, 2010. Print.

[14] A. Jacobs. "Statistical Analysis of Newspaper Headlines with Optimization." Web. <*http://repository.tudelft.nl/view/ir/*>.

[15] J. Friedman, T. Hastie, and R. Tibshirani. "Regularization Paths for Generalized Linear Models via Coordinate Descent." *Journal of Statistical Software* 33.1 (2010). Web. <*www.jstatsoft.org*>.

[16] R. Vanderbei. *Linear Programming Foundations and Extensions, 3rd ed.* New York: Springer, 2008. Print.

[17] J. D. Cook. "Cosines and Correlation." Web. <*http://www.johndcook.com/blog/2010/06/17/covariance-and-law-of-cosines/*>.