Enabling the Discovery of Recurring Anomalies in Aerospace Problem Reports using High-Dimensional Clustering Techniques

Ashok N. Srivastava, Ph.D., NASA Ames Research Center, Ashok.N.Srivastava@nasa.gov,
Ram Akella, Ph.D., University of California, Santa Cruz, akella@soe.ucsc.edu,
Vesselin Diev, University of California, Santa Cruz, vdiev@soe.ucsc.edu,
Sakthi Preethi Kumaresan, University of California, Santa Cruz, shakthi@soe.ucsc.edu,
Dawn M. McIntosh, NASA Ames Research Center, Dawn.M.McIntosh@nasa.gov,
Emmanuel D. Pontikakis, University of California, Santa Cruz, manos@stanford.edu
Zuobing Xu, University of California, Santa Cruz, zbxu@soe.ucsc.edu,
Yi Zhang, Ph.D., University of California, Santa Cruz, yiz@cmu.edu

Abstract— This paper describes the results of a significant research and development effort conducted at NASA Ames Research Center to develop new text mining algorithms to discover anomalies in free-text reports regarding system health and safety of two aerospace systems. We discuss two problems of significant import in the aviation industry. The first problem is that of automatic anomaly discovery concerning an aerospace system through the analysis of tens of thousands of free-text problem reports that are written about the system. The second problem that we address is that of automatic discovery of recurring anomalies, i.e., anomalies that may be described in different ways by different authors, at varying times and under varying conditions, but that are truly about the same part of the system. The intent of recurring anomaly identification is to determine project or system weakness or high-risk issues. The discovery of recurring anomalies is a key goal in building safe, reliable, and cost-effective aerospace systems.

We address the anomaly discovery problem on thousands of free-text reports using two strategies: (1) as an unsupervised learning problem where an algorithm takes free-text reports as input and automatically groups them into different bins, where each bin corresponds to a different unknown anomaly category; and (2) as a supervised learning problem where the algorithm classifies the free-text reports into one of a number of known anomaly categories. We then discuss the application of these methods to the problem of discovering recurring anomalies. In fact, because recurring anomalies tend to have very small cluster sizes, we explore new methods and measures to enhance the original approach for anomaly detection.

We present our results on the identification of recurring anomalies in problem reports concerning two aerospace systems as well as benchmark data sets that are widely used in the field of text mining. The first system is the Aviation Safety Reporting System (ASRS) database, which contains several hundred-thousand free text reports filed by commercial pilots concerning safety issues on commercial airlines. The second aerospace system we analyze is the NASA Space Shuttle problem reports as represented in the CARS data set, which consists of 7440 NASA Shuttle problem reports. We show significant classification accuracies on both of these systems as well as compare our results with reports classified into anomaly categories by field experts.[1]

TABLE OF CONTENTS

[1] 0-7803-9546-8//06//$20.00©2006 IEEE. This paper has been accepted for publication in the 2006 IEEE Aerospace Conference.

1. INTRODUCTION

There is an enormous amount of information regarding aerospace systems in the form of structured and unstructured text documents, much of it specifically relating to reports of anomalous behavior of craft, craft subsystem(s), and/or crew. Mining these text databases can result in the discovery of valuable information regarding the health of the overall system as well as specific indicators of the health of particular elements in the system.

There are at least two possible approaches to identifying anomalies in textual documents. The first approach, based on supervised learning, uses a mathematical model to 'learn' the relationship between a set of documents and some known target field. This target field, for example, could be a particular set of anomaly codes that have been assigned by human experts. This process can be described as a learning problem where the objective is to learn a function Φ that takes documents from a corpus $\mathcal{X}$ and maps them into an anomaly class $\mathbf{y}$, $\Phi(\mathbf{x}) \mapsto \mathbf{y}$. We optimize this function (also called a classifier) so that the number of errors that it makes is as small as possible given the training data $\mathcal{X}$.

The second approach is one where we do not have a predefined set of anomaly categories. In this situation, we need to develop another function Ψ that takes the document corpus $\mathcal{X}$ and divides it into a number of different categories automatically. This is an example of unsupervised learning, because the target is not predefined. This process has an infinite number of solutions, so the problem that we face is choosing one of those solutions that best matches the specific problem. We use clustering techniques, which have different statistical assumptions built into them, to learn a good mapping operator Ψ.

Clustering and classification techniques can be applied to group this large amount of data into known categories. In this direction, content based clustering of these reports helps detect recurring anomalies and relations in problem reports that indicate larger systemic problems. Clustering and classification methods and results will be presented using the Aviation Safety Reporting System (ASRS) database. The clustering results for two standard publicly available data sets will also be shown to allow method comparison to be performed by others.

The second problem addressed in this paper is to then autonomously identify recurring anomalies. The initial inroads for this work are described in detail in [2]. This approach will be presented and results shown for the CARS data set.

2. DATASETS USED

We have experimented with several standard data sets used for text classification. A brief description of each follows.

The 20 News Groups data set: This benchmark data set is a collection of 19997 documents belonging to 20 different news groups. Since the documents in this data set are primarily email messages, headers, such as from, to, subject, organization, etc. were removed in the preprocessing step. We were interested only in the body of the messages to keep it a free text classification exercise. We had an extensive stop word list, which was also removed from the documents. We tried to eliminate as many special characters as possible in order not to skew the results of the clustering algorithm.

The **Diff3 and Sim3 data sets** were created from the 20 News Groups data set to verify the performance of the algorithm in data sets that have closely related categories and dissimilar categories respectively.

The Reuters data set: This data set is the most widely used in text categorization research. It is a collection of 21578 documents each belonging to multiple classes.

Along with these standard data sets, we also tested our algorithms on real-world data sets regarding aerospace systems.

The CARS Data set: This data set is a collection of problem reports generated by engineers in different fields for the problems related to the NASA Space Shuttle. It contains a total of 7440 reports. Most of the reports are one or two short paragraphs. This data set was used for both text clustering and recurring anomaly algorithm development. Subject matter experts categorized 1553 of the 7440 reports into 366 recurring anomaly categories, providing a standard against which to measure the performance of the recurring anomaly algorithm. Consequently, there are $7440-1553 = 5887$ single document clusters, which make this problem suitable for clustering. Many of the 366 categories contained only two documents. None of the reports were clustered into more than one category.

The ASRS data set: After each commercial flight in the United States, a report is written for that flight describing how the flight went and whether any anomalous events

occurred. This data set is a collection of 20,696 of those reports categorized into 62 different anomalies. Between zero and twelve different anomalies were assigned to each report. The anomalies are labeled anomaly 413 through 474. Table 1 lists a subset of the 62 anomalies which were used for much of the analysis described in this paper.

Table 1. Table of Anomaly Labels and Anomaly Descriptions

Anomaly Label	*Anomaly Description*
413	Aircraft Equipment Problem - Critical
417	Other Spatial Deviation - Track or Heading Deviation
419	Airspace Violation - Entry
421	Altitude Deviation - Overshoot
430	Incursion - Runway
447	Inflight Encounter - Turbulence
450	Inflight Encounter - Weather
451	Inflight Encounter - VFR in IMC
453	Maintenance Problem - Improper Maintenance
462	Non Adherence - Clearance
466	Non Adherence - Required Legal Separation
468	Other Anomaly - Loss of Aircraft Control

3. Representing Text Documents in a Vector Space

The vector space model is a classic way of representing text documents. A database of text documents can be represented in the form of a Bag Of Words (BOW) matrix. Each row of the BOW matrix represents a document and each column represents a word from one or more documents. Therefore, the columns of the BOW matrix are the union of all of the words in all of the documents. Each word is associated with a Term Frequency (TF), which is given by the total number of times a word occurs in the document. Document Frequency is defined as the total number of documents in which the word w_i occurs. The $(\mathbf{i},\mathbf{j})\mathbf{th}$ cell of the BOW matrix corresponds to the TFIDF, which is the Term Frequency Inverse Document Frequency of the $\mathbf{jth}$ word in the document. The TFIDF is defined as: $TFIDF = TF.IDF$, where $IDF(w_i) = log(n/DF(w_i))$, where n is the total number of documents in the document database. Thus each text document is represented as a point in a high dimensional vector space. Since the number of terms in the union of all documents is likely to be very large, the BOW matrix is high dimensional. In some applications that we have studied, the dimension can be over 30,000. A variety of techniques like Principle Component Analysis (PCA), Singular Value Decomposition (SVD) and Information Theoretic approaches have been used to reduce the dimensionality of the vector space. [2]

4. A Brief Look at Clustering Methods

A wide variety of methods in the field of machine learning have been used to cluster text documents. In [3], Joachims claims that most text categorization problems are linearly separable, making them ideal candidates for Support Vector Machines (SVMs). In [4], he makes an attempt to bring out the statistical similarity between the parametric and non-parametric approaches for clustering.

The non-parametric methods in the classification of text documents are generally algorithms like K-means and Nearest Neighbor classification. Consider a set of data points distributed in a d dimensional space. K-means chooses a set of initial points as the seeds. In step one, each document in the data set is associated with that seed document to which it has the minimum Euclidean distance. This results in the classification of documents into k clusters. In step two, the seed associated with each cluster is updated to the mean of all document vectors in that particular cluster. With the updated seeds, step one is repeated again and the process continues iteratively. Some of the documents get assigned to different clusters and the seeds are repeatedly updated. The algorithm converges when either the update on the seeds is ignorable or the documents are no longer assigned to different clusters during each iteration. In the following paragraphs we will describe this algorithm in the context of the Gaussian Mixture Model.

Mixture models can loosely be classified as a group of methods that involve parameter estimation. Consider a mixture of distributions from the exponential family. The underlying random variable could be generated from any one of the distributions in the mixture model with a probability equal to the prior probability associated with that particular distribution.

The Gaussian Mixture Model assumes that the text documents were generated using a mixture of k gaussian dis-

tributions, each with its own parameters $\theta_\mathbf{i}$

$$\sum_{i=1}^{k} \alpha_i f(x|\theta_i) \tag{1}$$

such that $\sum_i \alpha_i = 1$, where α_i is the prior probability of the ith distribution in the mixture model. Each probability distribution function is representative of a particular category of documents. If there are k categories in a document database, then this situation can be typically modeled using a mixture model of k distributions.

Expectation Maximization Algorithm and its Application to Text Classification

We will explain the Expectation Maximization (EM) algorithm in the framework of the mixture model. [5] It is an iterative approach to calculate the parameters of the mixture model mentioned in the previous section. It consists of two steps: The Expectation step or E-step and the Maximization step or the M-step. In the E-step, the likelihood that the documents were generated using a given distribution in the mixture model is estimated. Each document is assigned to that cluster whose representative probability density function has the highest likelihood for generating the document. This results in the classification of documents into one of the n classes, each represented by a particular probability density function. In the M-step, the maximum likelihood estimates of the parameters of each distribution is calculated. This step uses the classification results of the E-step, where each class is assigned a set of documents. We will attempt to explain the E-step and M-step in the context of the Gaussian Mixture Model. Let us assume that we have M data points that we want to model using a mixture of K univariate Gaussian distributions with identical and known variance. The unknowns here are the parameters of the K gaussian distributions. Also the information on which data point was generated using which of the distributions in the mixture is unknown. Each data point Y_m is associated with $\mathbf{K}$ hidden variables $\{w_{m,1}, w_{m,2}, w_{m,3}, \ldots, w_{m,k}\}$ where $w_{m,k} = 1$, if Y_m was generated using distribution k, otherwise $w_{m,k} = 0$. The maximum likelihood (ML) estimate of the mean μ_k of the kth distribution is given by,

$$\mu_k = \frac{1}{M_k} \sum_{m=1}^{M} w_{m,k} Y_m \tag{2}$$

where $M_k = \sum_{m=1}^{K} w_{m,k}$

The problem is that we know neither the value of μ_k nor the hidden variables $w_{m,k}$.

E step: The expected values of the $w_{m,k}$ are calculated based on the current estimates of the Gaussian parameters μ_k.

$$\begin{aligned} E(w_{m,k}) &= p(k|Y_m) \\ &= \frac{p(Y_m|k)p(k)}{p(Y_m)} \\ &= \frac{p(Y_m, k)}{p(Y_m)} \\ &= \frac{\exp \frac{-(Y_m-\mu_k)^2}{2\sigma^2}}{\sum_{j=1}^{K} \exp \frac{-(Y_m-\mu_j)^2}{2\sigma^2}} \end{aligned} \tag{3}$$

This corresponds to clustering data points by minimizing the Euclidean distances in the k-means algorithm.

M step: Using the expected values of $w_{m,k}$ the ML estimates of μ_k are calculated. This corresponds to updating the seeds of clusters centers at every iteration of the k-means algorithm. Or, in other words, the M step corresponds to recalculating the seeds of the k-means algorithm. The center of the cluster corresponds to the mean of all the documents or data points in the corresponding cluster.

Thus the k-means algorithm could be explained as a solution to estimating the parameters of the Gaussian Mixture Model. The parameters are determined by maximizing the expected value of the log likelihood function. However in the context of text clustering, the high dimensional document vectors used to represent text documents are very sparse and the algorithm, does not work on sparsely located data points in a high dimensional space. Hence we will explore mixture models that can provide more effective representation and analysis of text documents.

5. DIRECTIONAL STATISTICS

Directional statistics is a field of statistics that analyzes the statistical properties of directional random variables. For example, a random variable, θ, representing the position of a roulette wheel can be said to exhibit directional statistics.

The preprocessing step before applying the clustering algorithms to text data involves normalization. The TFIDF document vectors are L_2 normalized to make them unit norm. Here the assumption is that the direction of documents is sufficient to get good classification and hence by normalization, the effect of the length of the documents is nullified. For example, two documents - one

short, one long - on the same topic will have the same direction and hence will be put into the same cluster. If the dimension of the vector space before normalization is $\mathbf{R^d}$, the unit normalized data lives on a sphere in an $\mathbf{R^{d-1}}$ dimensional space. Hence, it is more appropriate to use directional distributions to model this data.

The von Mises Fisher Distribution

The Von Mises Fisher (VMF) distribution is one of the directional distributions. It was developed by von Mises to study the deviations of measured atomic weights from integer values. Its importance in statistical inference on a circle is almost the same as that of the normal distribution on a line.

A circular random variable, θ, is said to follow a von Mises distribution if its prior distribution function (pdf) is given by:

$$g(\theta; \mu_o, \kappa) = \frac{I}{2\pi I_o(\kappa)} \exp \kappa \cos(\theta - \mu_o), \quad 0 \leq \theta \leq 2\pi, \kappa > 0, 0 \leq \mu_o \leq 2\pi \quad (4)$$

where $I_o(\kappa)$ is the modified Bessel function of the first kind and order zero. The parameter μ_o is the mean direction while the parameter κ is described as the concentration parameter. A unit random vector $\mathbf{x}$ is said to have d variate von Mises-Fisher distribution if its pdf is:

$$c_d(\kappa) e^{\kappa \boldsymbol{\mu}^T \mathbf{x}} dS^{d-1}, \qquad \mathbf{x} \in \mathbf{S^{d-1}} \subseteq \Re^{\mathbf{d}} \quad (5)$$

where $\parallel \mu \parallel$ and $\kappa \geq 0$. The closed form expression for κ is given by:

$$C_p(\kappa) = \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{\frac{d}{2}-1} \kappa} \quad (6)$$

The Choice of VMF among all other Spherical Distributions

This section analyzes the appropriateness of using the von Mises distribution for text classification among all other spherical distributions. Is there a Central Limit Theorem (CLT) for directional data? Does it correspond to the CLT for non-directional data? For data on a line, the CLT says that the normal distribution is the limiting distribution. Whereas for directional data, the limiting distribution of the sum of n independent random variables is given by the uniform distribution. In spite of this, the uniform distribution is not a contender for modeling directional data [6].

Relation to bivariate normal distribution: The VMF shows several analogies to the properties of the normal distribution. Due to space limitations, we will discuss briefly a few such analogies. Maximum Likelihood Characterization: Consider the distribution of a random variable on the real line. Let $f(x - \mu)$ represent the distribution where μ is the mean. The maximum likelihood estimate for μ is given by the sample mean if and only if the distribution is Gaussian. Similarly, for a random variable θ on a circle, let the directional distribution be given by $g(\theta - \mu_o)$. The Maximum Likelihood estimate for the mean μ_o is given by the sample mean $\mathbf{x_o}$, if and only if the directional distribution is the VMF distribution. Maximum Entropy Characterization: Given a fixed mean and variance for a random variable x, the Gaussian is the distribution that maximizes the entropy. Likewise for a circular random variable, θ, given a fixed mean direction μ_o and circular variance, the VMF distribution maximizes the entropy.

Unfortunately there is no distribution for directional data which has all properties analogous to the normal distribution. The VMF has some but not all of the desirable properties. The wrapped normal distribution is a strong competitor to VMF. But the VMF provides simpler ML estimates. Also the VMF is more tractable while doing hypothesis testing. Hence the use of VMF over other directional distributions appears well warranted.

6. The VMF Algorithm for Clustering

In this section we discuss the theory behind modeling the text documents using a mixture model of VMF distributions. Consider a mixture model consisting of K VMF distributions similar to Equation 1. Each distribution is attributed a prior probability of α_k with $\sum_{k=1}^{k} \alpha_k = 1$ and $\alpha_k \geq 0$. It is given by:

$$f(\mathbf{x}|\mathbf{\Theta}) = \sum_{\mathbf{k=1}}^{\mathbf{K}} \alpha_{\mathbf{k}} \mathbf{f_k}(\mathbf{x}|\theta_{\mathbf{k}}) \quad (7)$$

Here $\Theta = \{\alpha_1, \alpha_2, \ldots, \alpha_k, \theta_1, \theta_2, \ldots, \theta_k\}$. $\theta_k = (\boldsymbol{\mu}, \kappa)$. Let $Z = \{z_1, ...z_N\}$ be the hidden variables associated with the document vectors $\mathbf{X}\{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_N}\}$. $z_i = k$, if the document vector $\mathbf{x_i}$ was generated from the kth VMF distribution. Assuming that the distribution of the hidden variables $p(k|\mathbf{x}, \mathbf{\Theta}) = \mathbf{p}(\mathbf{z_i} = \mathbf{k}|\mathbf{x} = \mathbf{x_i}, \mathbf{\Theta})$ is known, the complete log likelihood of the data, with expectation taken over the distribution p, is given by:

$$E_p[\ln P(\mathbf{X}, \mathbf{Z}|\mathbf{\Theta})] = \sum_{k=1}^{K}\sum_{i=1}^{N} \ln \alpha_k p(\mathbf{x_i}|\mathbf{\Theta}) + \sum_{k=1}^{K}\sum_{i=1}^{N} (\ln f_k(x_i|\theta_k)) p(k|\mathbf{x_i}, \mathbf{\Theta}). \quad (8)$$

The Maximization Step: In the maximization step, we estimate Θ by maximizing (8). By taking partial derivatives of (8) w.r.t the parameters, the ML estimates are given by:

$$\hat{\alpha}_k = \frac{1}{N}\sum_{i=1}^{N} p(k|\mathbf{x_i}, \mathbf{\Theta}) \quad (9)$$

$$\hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^{N} \mathbf{x_i} \mathbf{p}(\mathbf{k}|\mathbf{X}, \mathbf{\Theta})}{\| \sum_{i=1}^{N} \mathbf{x_i} \mathbf{p}(\mathbf{k}|\mathbf{X}, \mathbf{\Theta}) \|} \quad (10)$$

The ML update for κ, obtained after approximations is given by:

$$\hat{\kappa}_k = \frac{\bar{\mathbf{r}}_k d - \bar{\mathbf{r}}_k^3}{1 - \bar{\mathbf{r}}_k^2} \quad (11)$$

where $\mathbf{r_k} = \frac{\mathbf{I_{d/2}}(\kappa)}{\mathbf{I_{d/2-1}}(\kappa)}$

The Expectation Step: Assuming that the ML updates calculated from the above step are right, the expectation step updates the distribution of the hidden variables Z. There are two ways of assigning the documents to clusters: the soft and hard assignments. The distribution of the hidden variables as considered in the soft assignment scheme:

$$p(k|\mathbf{x_i}, \mathbf{\Theta}) = \frac{\alpha_\mathbf{k} \mathbf{f_k}(\mathbf{x_i}|\mathbf{\Theta})}{\sum_{\mathbf{k=1}}^{\mathbf{K}} \alpha_\mathbf{k} \mathbf{f_k}(\mathbf{x_i}|\mathbf{\Theta})} \quad (12)$$

Under the hard assignment scheme, the update equations are given by:

$$q(k|\mathbf{x_i}, \mathbf{\Theta}) = \begin{array}{ll} 1 & \text{if } k = argmax_{k'} q(k'|\mathbf{x_i}, \mathbf{\Theta}) \\ 0, & \text{otherwise} \end{array} \quad (13)$$

So according to (13), the documents either belong to a cluster or they do not. There is no notion of the documents belonging to several clusters. There is no *one to many* mapping between the document and cluster domains. In practice this may be disadvantageous because some data sets like the Reuters data set have multi-labeled documents. A few of the most popular classes in the Reuters data set are ACQ, CORN, WHEAT and EARN. In this case, there are documents that belong to ACQ, EARN and WHEAT. It would be impossible to get this kind of categorization using the hard assignment scheme.

Although the update equations for the VMF algorithm derived in the previous section have closed form expressions, when the dimensionality of the vector space expands, the computations increase. Typical bag of words (BOW) matrices are of the order of $10^4 \times 10^4$. Since the dimension of the document vectors is of the order of 10^4, the update equations for κ in (11), require the calculation of very high order Bessel functions. Owing to this, the computational time of the algorithm increases and this also resulted in precision problems while tracking the variables through the iterations. To overcome this problem, mathematical approximations were used in the update equations. For a modified Bessel function of the first kind and order $\mathbf{n}$, for large $\mathbf{x}$, fixed $\mathbf{n}$ and $\mathbf{x} >> \mathbf{n}$, the approximation is given as follows:

$$I_n(x) \sim \frac{e^x}{\sqrt{2\pi x}} \quad (14)$$

When tested over several data sets, imputing this approximation gave better or equally good performance, compared to using the exact expression for Bessel functions.

7. Simulation Results

We present the results of our unsupervised clustering methods on two benchmark data sets. Although these data sets are not related to the aerospace industry, they allow us to test our algorithms. Subsequently, we test our algorithms on the Aviation Safety Reporting System (ASRS) data set which is a well-known textual data set for aviation safety. Mutual Information is used as the criteria for comparing the performance of the different methods on the various data sets. Mutual Information is generally used in statistics to measure the degree of information that be obtained about one random variable by knowing the value of another random variable. Consider two random variables x and y. Let $p(x)$ and $p(y)$ be the marginal distributions of x and y and let the joint distribution be $p(x, y)$. The Mutual Information between x and y is defined as:

$$I(X;Y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \quad (15)$$

We used the Mutual Information between the vector of class labels produced by the algorithms and the actual class labels of the documents as the criterion to compare the performance of the different algorithms. We compare the performance of the VMF algorithm against spherical K-means (spk-means) algorithms. spk-means is a modified version of k-means. The cosine of the angle between two data vectors points is used the measure of similarity in spk-means in contrast to the Euclidean distance used in the case of k-means.

20 Newsgroups Data Set Results

Figure 1 shows the performance of VMF versus spherical K-means on the 20 News Groups diff3 data set. It can be seen that the performance of the VMF algorithm is significantly better for this data set. In Figure 2, the performance curves for the VMF and spk-means algorithms on the sim3 data set are shown. These results were originally discussed in [1].

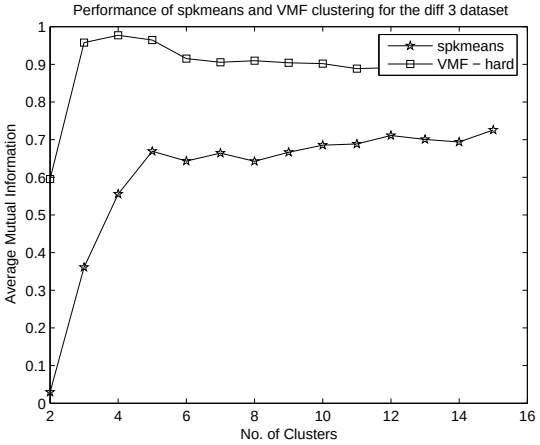

Figure 1. This figure shows a comparison between the performance of the VMF and spherical K-means algorithms for the diff3 Data Set, which is a data set where the categories have very little overlap. The x-axis shows the number of clusters and the y-axis shows the mutual information averaged over 20 runs. Notice that the mutual information is as much as 30% higher for the VMF algorithm when compared to spherical k-means.

Reuters Data Set Results

The fact that the top frequency words in a cluster are representative of the subject of the cluster is depicted in Figure 3. This figure represents the top frequency words from two clusters CRUDE OIL and GRAIN from the Reuters data set. The first row represents the distribution

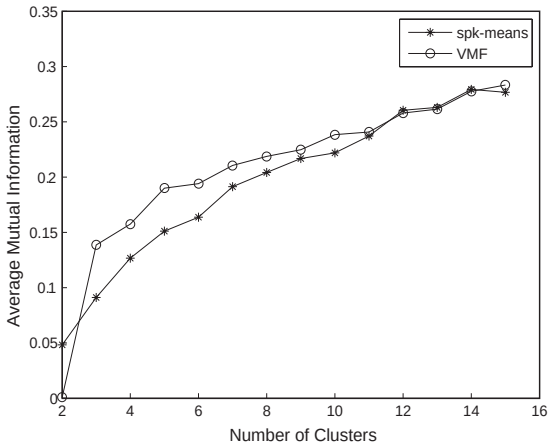

Figure 2. Comparing the Performance of VMF and spherical K-means algorithms for the 20 News Groups Sim3 data set. This data set contains documents that have a high degree of similarity between them, thus complicating the clustering task. The VMF algorithm slightly outperforms the spherical k-means algorithm on this benchmark data set.

of the first four most frequently occurring words in cluster one. It compares the distribution of each word across the whole dataset to the distribution across cluster one. It can be seen these words occur with a higher probability in the documents of cluster one. Thus the words oil, price, and barrel indicate that cluster one contains documents belonging to category, *crude oil*. Similarly, cluster two contains documents belonging to category *grain*. The word, mln, is probably an abbreviation that occurs in most documents belonging to both clusters one and two. If it could be identified as a stopword and removed from the dataset, the performance of the algorithm would improve.

8. Recurring Anomaly Detection

We shift our attention to the problem of detecting recurring anomalies in text documents. Recurring anomaly detection is also an unsupervised learning problem. The task of recurring anomaly detection has not been addressed by prior work, because of the unique structure of the problem. The research most closely related to recurring anomaly detection is perhaps the Novelty and Redundancy Detection in Adaptive Filtering.[13] Novelty and redundancy detection distinguishes among relevant documents that contain new (novel) information and relevant documents that do not. The definition of recur-

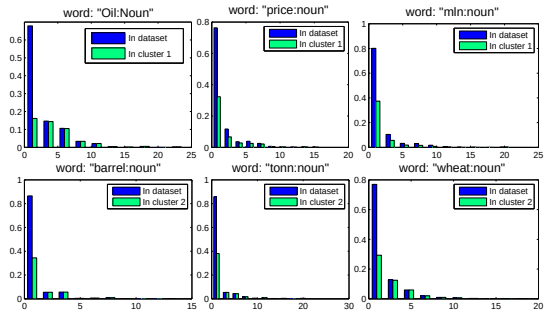

Figure 3. Word distribution plots of the Reuters Data set for cluster 1 and cluster 2. Note the cluster 1 terms relate to CRUDE, and the cluster 2 terms relate to GRAIN. These plots show that the frequency of occurrence of key words significantly affects the clustering results.

ring anomaly in our problem matches the definition of redundancy. The difference between them lies in two aspects: 1. Novelty and Redundancy Detection processes the documents in sequence, and recurring anomaly detection does not. 2. Recurring anomaly detection groups recurring anomalies into clusters, and Novelty Detection does not. Another research field related to recurring anomaly detection is the retrospective event detection task in Topic Detection and Tracking [14] [15]. The retrospective event detection task is defined to be the task of identifying all of the events in a corpus. Recurring anomaly detection task differs from their task in having many single document clusters. However, the similarity of the tasks is worth exploring, and several methods we investigated are motivated by their work. The core part of our work concerns similarity measures between statistical distributions. A complete study on distributional similarity measures is presented by [16].

Language Models and Similarity Measures

There are two general approaches to measure the similarity between documents: non-statistical methods and statistical methods. One of the typical non-statistical methods is cosine distance, which is a symmetric measure related to the angle between two vectors. It is essentially the inner product of the normalized document vectors. If we present document d as a vector $d = (w_1(d), w_2(d), \ldots, w_n(d))^T$, where $w_i(d)$ is the number of occurrence of word i in document d, then the similarity between two documents d_t and d_j can be represented by:

$$cos(d_t, d_j) = \frac{\sum_{k=1}^{n} w_k(d_t) w_k(d_j)}{\|d_t\| \, \|d_j\|} \qquad (16)$$

The statistical language model used in most previous work is the unigram model. This is the multinomial model which assigns the probability of the occurrence of each word in the document

$$P(d) = \prod_{w_i} p(w_i, d)^{tf(w_i, d)} \qquad (17)$$

where $p(w_i, d)$ is the probability that word i occurred in document d, and $tf(w_i, d)$ indicates how many times word i occurred in the documents.

We assume each document is generated by a distinct multinomial distribution, and we discover and cluster recurring anomalies based on the distance between different distributions. In contrast, in the mixture model we used in the previous section, each distribution generates a cluster of documents. In the recurring anomaly detection problem described here, there are many single document clusters. In a statistical sense single document cluster is a single sample generated by the underlying distribution. The reason that we do not use von Mises Fisher distribution, which we used in the previous section, is that we can not estimate the mean and the variance unless we have a enough data points (documents). If we attempt to use a VMF distribution to model a set of clusters, where many clusters contain a single document, we run into the following difficulty: each single document cluster gives us zero variance and mean value which corresponds to that of the single document vector itself; this is not very meaningful in our context.

The problem is reduced to a multinomial distribution parameter estimation problem. The maximum likelihood estimation of the probability of a word occurring in the document is shown below as :

$$p(w_i|d) = \frac{tf(w_i, d)}{\sum_{w_j} tf(w_j, d)} \qquad (18)$$

Furthermore, we use an algorithm based on generative model of document creation. This new mixture word model measure is based on a novel view of how relevant documents are generated. We assume each recurring anomaly document is generated by the mixture of three language models: a general English language model, a topic model, and a document-specific information model. For instance, in a short document ”the airplane engine has some electric problems,” the words ”the,” ”has” and ”some” probably come from the general English model, words such as ”airplane” and ”problem” are likely generated from the topic model, and the words ”engine” and ”electric” are generated from the

document-specific information model. Because all the documents are anomaly reports on airplanes, documents are likely to contain words like "airplane" and "problem." The information contained in the document specific model is useful to detect recurring anomalies caused by different problems. So measuring the similarity only between the document specific models makes the recurring anomaly detection more accurate. Each word is generated by each of the three language models θ_E, θ_T and θ_{dcore} with probability λ_E, λ_T and λ_{dcore} respectively:

$$P(w_i|\theta_E, \theta_T, \theta_{dcore}, \lambda_E, \lambda_T, \lambda_{dcore}) = \lambda_E P(w_i|\theta_E) + \lambda_T P(w_i|\theta_T) + \lambda_{dcore} P(w_i|\theta_{dcore}) \quad (19)$$

where λ_E, λ_T and λ_{dcore} are the probabilities of a word generated by general english language model, topic model and document specific model, respectively, and $\lambda_E + \lambda_T + \lambda_{dcore} = 1$. If we fix λ_E,λ_T and λ_{d_core}, then there exists a unique optimal value for the parameters of document core model that maximizes the likelihood of the document. We employ a quick algorithm based on Lagrange multiplier method to find the exact optimal solution, given fixed mixture weights λ_E, λ_T and λ_{dcore} [17].

Kullback-Leibler divergence, a distributional similarity measure, is one way to measure the distance between two multinomial distributions.

$$KL(\theta_{d_t}, \theta_{d_j}) = \sum_{w_i} p(w_i)|\theta_{d_t} \log(\frac{p(w_i|\theta_{d_t})}{p(w_i|\theta_{d_j})}) \quad (20)$$

where θ_{d_t} and θ_{d_j} are the parameters of two multinomial distributions, which can be estimated either using general maximum liklihood estimation (18) or mixture word model (19). The problem with KL divergence is that if a word, w_i never occurs in a document, it will get a zero probability $p(w_i|d) = 0$. Thus a word not in d_t but in d_j will cause $KL(\theta_{d_t}, \theta_{d_j}) = \infty$. To avoid the singularity of KL divergence, we resort to other measurements: Jensen-Shannon divergence, Jaccard's Coefficient and skew divergence. Jensen-Shannon divergence [16] has proven to be a useful symmetric measure of the distance between distributions

$$JS(\theta_{d_t}, \theta_{d_j}) = \frac{1}{2}[KL(\theta_{d_t}, avg_{d_t,d_j}) + KL(\theta_{d_j}, avg_{d_t,d_j})] \quad (21)$$

where $avg_{d_t,d_j} = 1/2(\theta_{d_t} + \theta_{d_j})$.

We also use skew divergence [16] to measure the similarity between two discrete distributions. Skew divergence is an asymmetric generalization of the KL divergence,

$$Sk(\theta_{d_t}, \theta_{d_j}) = KL(\theta_{d_t}, (1-\alpha)\theta_{d_t} + \alpha\theta_{d_i}),\ 0 \le \alpha \le 1 \quad (22)$$

Note that at $\alpha = 1$, the skew divergence is exactly the KL divergence, and at $\alpha = 0.5$, the skew divergence is twice one of the summands of Jensen-Shannon divergence. In our experiment, we choose $\alpha = 0.99$ to approximate the KL divergence and avoid singularity.

Jaccard's coefficient differs from all the other measures, being essentially based only on the sizes of the supports of the document specific distribution rather than the actual value of the distribution

$$Jac(\theta_{d_t}, \theta_{d_i}) = \frac{\{v : \theta_{d_t}(v) > 0 \text{ and } \theta_{d_j}(v) > 0\}}{\{v : \theta_{d_t}(v) > 0 \text{ or } \theta_{d_j}(v) > 0\}}. \quad (23)$$

Based on the similarity measurement between anomaly documents, we apply an agglomerative hierarchical clustering method to partition the documents. The agglomerative hierarchical algorithm produces a binary tree of clusters in a bottom-up fashion: the leaf nodes tree are single document clusters; the middle-level node is the centroid of the two most proximate lower level clusters; and the root node of the tree is the universal cluster which contains all the documents. The agglomerative hierarchical clustering method we apply is single linkage clustering. For single linkage clustering, the linkage function specifying the distance between two clusters is computed as the minimal object-to-object distance. In other words, the distance between two clusters is computed as the distance between the two closest objects in the two clusters. In order to cluster the data points, we specify a threshold on the similarity and if the distance between two data points is less than the threshold, we cluster these two documents together.

New Performance Measures for Recurring Anomalies

The recurring anomaly detection problem can be decomposed into two parts: discovering recurring anomalies and clustering recurring anomalies, so there is a need for different performance measures. Now we present a simple example to indicate the need for the new performance measure.

Suppose we only have 10 anomaly documents. In the column "Algorithm" in Table 2, we see that our algorithm groups the documents into 5 clusters; the column "Expert" shows the expert clustering results. The bold numbers in each column correspond to the recur-

ring anomalies detected by the algorithm and recurring anomalies labeled by the expert, respectively.

Table 2. Simple clustering example for illustrating new performance measure

	Algorithm	Expert
Cluster 1	**1**,**2**,**5**,**6**	**1**,**2**,**3**,**4**
Cluster 2	**3**,**4**,**7**	**5**,**8**
Cluster 3	9	**9**,**10**
Cluster 4	10	6
Cluster 5	8	7

In this example the algorithm has made the following errors: it missed recurring anomaly 8 in expert cluster 2; incorrectly categorized anomaly documents 6 and 7 as recurring anomalies; incorrectly separated expert anomaly cluster 1, which contains anomaly documents 1, 2, 3 and 4, into two clusters; incorrectly separated expert anomaly cluster 3, which contains anomaly documents 9 and 10, into two single clusters; and incorrectly combined recurring anomaly documents 1, 2, 5 and non-recurring anomaly document 6 into one cluster. So we summarize the errors into four categories: 1. miss a recurring anomaly, 2. incorrectly indicate a non-recurring anomaly, 3. incorrectly separate same category of recurring anomalies into different clusters, 4. incorrectly combine different kinds of recurring anomalies into one cluster. We use R^+, R^-, N^+, and N^- to denote the number of documents that fall into the following categories (Table 3) We apply the standard precision (purity

Table 3. Precision and recall categories

	Labeled by Expert as recurring anomaly	Not labeled by Expert as recurring anomaly
Detected as recurring anomaly	R^+	N^+
Not detected as recurring anomaly	R^-	N^-

of recurring anomaly detection) and recall (completeness of recurring anomaly detection) measure to characterize the first two errors as we described above.

$$Precision = \frac{R^+}{R^+ + N^+} \tag{24}$$

$$Recall = \frac{R^+}{R^+ + R^-} \tag{25}$$

Table 4. Same simple clustering example for illustrating the new performance measure (i.e., keep anomalies which are both detected by algorithm and and labeled by expert)

	Algorithm	Expert
$Cluster1$	1,2,5	1,2,3,4
$Cluster2$	3,4	5

In the given example, the number of recurring anomaly clusters which are both detected by the algorithm and labeled by the expert is 5; the total number of recurring anomaly documents detected by the algorithm is 7; the total number of recurring anomaly documents labeled by the expert is 8. Therefore, the precision is $5/7$ and the recall is $5/8$. Precision and recall here measure the accuracy of detecting recurring anomalies, but do not characterize the accuracy of clustering anomalies. Because only the recurring anomaly documents which have been both detected by the algorithm and labeled by the expert affect the accuracy of the clustering, we use these anomalies to calculate the miscombination and misseparation score. The remaining anomalies are shown in Table 4.

To measure the errors caused by incorrectly separating recurring anomalies, which are in the same expert cluster, into different clusters, we add up the reciprocal of the number of separated clusters corresponding to each expert cluster and normalize it by the total number of clusters in the expert result. If the algorithm result exactly matches the expert result, the score for that expert result is 1. The score decreases as the number of separated clusters increases. The counterpart of misseparation by the algorithm is miscombination of the expert categories, so we use the same scheme but based on the algorithm result to calculate the miscombination score. To express the misseparation score and miscombination

score metric, we use the following notation.

NSA_i = number of algorithm clusters which incorrectly separate the anomalies in the ith expert cluster

NSE_i = number of expert clusters which are incorrectly combined into the ith algorithm cluster

NE = total number of clusters in expert result

NA = total number of clusters in algorithm result

Therefore the misseparation and miscombination scores are defined as following,

$$Misseparation = \frac{\sum_{i=1}^{NE} \frac{1}{NSA_i}}{NE} \quad (26)$$

$$Miscombination = \frac{\sum_{i=1}^{NA} \frac{1}{NSE_i}}{NA} \quad (27)$$

To better understand the measure scheme, we will explain it with the same simple example, Table 2. The algorithm separated anomaly 1, 2, 3 and 4 which are in expert cluster 1, into two clusters, so the misseparation score for this cluster is 1/2; for expert cluster 2, since there is only one anomaly, the misseparation score is 1. The overall normalized score for misseparation is $\frac{1/2+1}{2} = 0.75$. The miscombination score can be calculated with the same approach.

Experimental Results and Conclusions

The aerospace system we analyzed is the NASA Space Shuttle problem reports as represented in the CARS data set, which consists of 7440 NASA Shuttle problem reports. We conducted experiments on the CARS data set to compare the performance of different similarity measure schemes.

To testify to the effectiveness of the mixture language model, we compared the performance of the skew divergence measure based on mixture language model (19) and the skew divergence measure based on general language model, which use maximum likelihood estimation (18). The results are shown in Figure 4 and Figure 5. From Figure 4, we see that the mixture model result is consistently more accurate than the general model. Figure 5 shows that the mixture language model and general

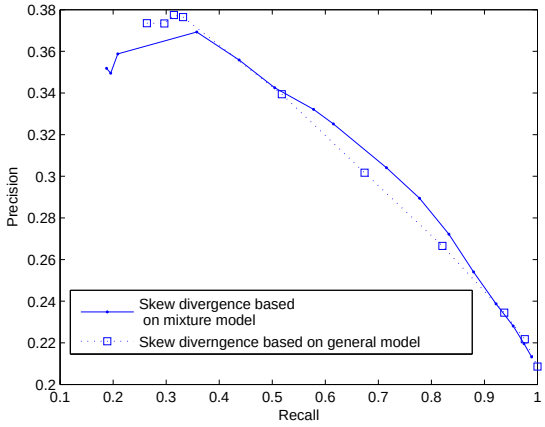

Figure 4. Precision and Recall Comparison between Skew Divergence based on Mixture language Model and Skew Divergence based on General Language Model

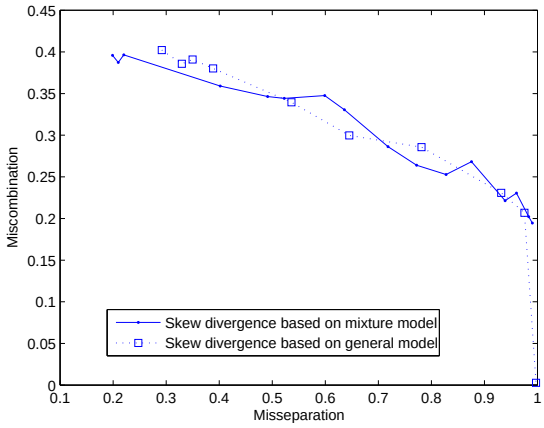

Figure 5. Misseparation and Miscombination Comparison between Skew Divergence based on Mixture language Model and Skew Divergence based on General Language Model

model are comparable. The experiments on verifying the mixture language model are also performed on the other statistical similarity measures, and results consistently show that the mixture language model outperforms the general language model.

We also compared the performance of four similarity measures: cosine distance, skew divergence based on mixture language model, Jenson-Shannon divergence based on mixture word model, and Jaccard's coefficient based on mixture word model. Here we compared the

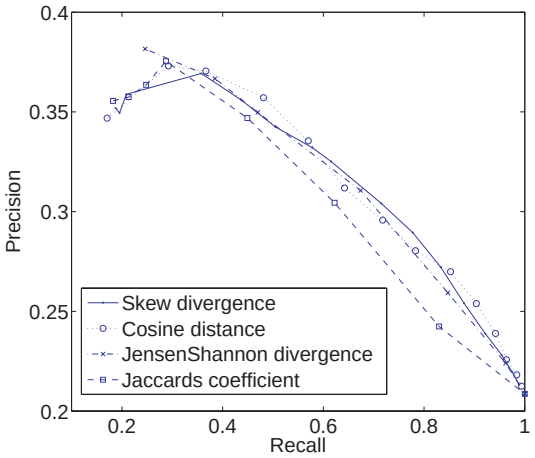

Figure 6. Precision and Recall comparison of Skew Divergence based on mixture language model, Jenson-Shannon Divergence based on mixture language model, Jaccard's Coefficient based on mixture language model and Cosine distance

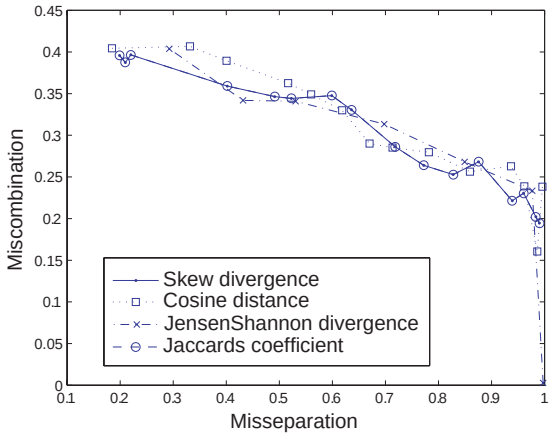

Figure 7. Misseparation and Miscombination comparison of Skew Divergence based on mixture language model, Jenson-Shannon Divergence based on mixture language model, Jaccard's Coefficient based on mixture language model and Cosine distance

statistical similarity measures based on the mixture language model with the non-statistical similarity measure, cosine distance, because previous experiments have already shown that the mixture language model is more effective than general language model.

Figure 6 and Figure 7 present two types of comparison measures for each method. Figure 6 shows that the skew divergence method based on the mixture language model and the cosine distance method are very effective. In general, they outperform the other two methods. The Jaccard's coefficient measure is the least accurate. There was value in doing this comparison because it showed that the traditional cosine similarity metric is very effective, even though cosine similarity is less well-justified statistically compared with language modeling approach. However, cosine similarity has been demonstrated many times and over many tasks to be a robust similarity metric. Our results add recurring anomaly detection to the long list for which it is effective. In the region, where the recall ranges from 0.55 to 0.85, the skew divergence is the most accurate. This region satisfies the user requirements: relatively high recall and low precision.

9. Text Classification of Aviation Safety Reports into Anomaly Categories

There are a number of predefined anomalies which can occur in the aircraft during a flight. The goal of text classification is to develop a system that based on the semantic meaning of a report infers which, if any, anomalies have occurred during a flight for which a report has been written. The work at the semantic level has already been done and we are given the resulting BOW matrix where the words are extracted by natural language processing (NLP) methods. As mentioned in a previous section, there are a total of 20,696 reports, a total of 28,138 distinct terms, and a total of 62 different anomalies. A report can have between 0 and 12 anomalies. Whether a particular anomaly has occurred or not is labeled by 1 and 0 respectively in the training data set. Most reports (over 90% of them) contain more than 1 anomaly, with the most common group of reports containing exactly 2 anomalies (5,048 reports). The most frequent anomaly occurs in almost half of the reports.

System Overview

By running association rules on the anomaly labels, it was determined that the correlation among different anomalies is not strong. Therefore each anomaly can be treated individually. Thus the multi-label classification problem is treated as a binary classification problem for

every anomaly. For this reason, and others, the VMF algorithm limitations prevent it from being the appropriate algorithm for analysis of the full data set. As an initial step, 12 of the 62 anomalies were selected with the goal of trying to find a classifier that will perform best for each of them.

This approach can be summarized in three main phases. In the first phase the data is loaded into a database, statistics are collected on it for the purposes of studying the data, then the terms with very low frequency are removed. In the second phase, common feature selection algorithms are run to reduce the feature space by picking the best terms for each anomaly. In the final phase, experiments were run with several commonly used text classification algorithms, such as Support Vector Machines (SVM), Naive Bayes, AdaBoost, Linear Discriminant Analysis (LDA), Logistic Regression, implemented in the open-source packages WEKA [7], SVM-light [8] and R. We show convincingly that SVM, with an Radial Basis Function (RBF) kernel in particular, performs best for this particular text classification problem. Figure 8 summarizes our architecture.

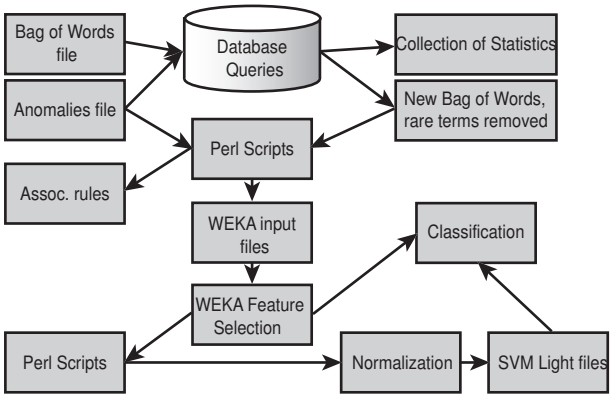

Figure 8. This block diagram shows the architecture for the text classification system that we built to classify Aviation Safety Reports. Data were loaded into a MySQL database and open-source data mining software was used for the analysis and classification of the data.

All terms which appear in exactly one report, regardless of their frequencies, are removed. The intuition behind this is that those terms are not frequent enough to be used for training and will most likely not be seen in the test data. Also, since even the low frequent anomalies occur in at least hundreds of reports, we do not expect much contribution of the rare terms to the classification problem. After the removal of those rare terms, the total number of terms left is 17,142.

In this phase we perform feature reduction by selecting the most informative terms for every anomaly [11][12]. The Information Gain (IG) criterion is used to rank the terms according to how informative they are for a specific anomaly:

$$IG(class, term) = H(class) - H(class|term) \quad (28)$$

where $H(class)$ denotes the entropy of a specific anomaly, and $H(class|term)$ denotes the conditional entropy of an anomaly given a particular term. For every anomaly, we experimentally find out which is the optimal number of terms. This is an iterative process and includes picking different numbers of best terms for each anomaly and then running several different classifiers and analyzing the performance results. For some anomalies, it is best to keep the top 1000 ranked terms out of 17,142 and for some others this number is 500 or 1500. For efficiency purposes we set 1500 as an upper threshold of the number of terms we would work with. Working with just the best 500, 1000, or 1500 terms for each anomaly helps speed up the classification process and at the same time increases the classification accuracy. It was observed that infrequent anomalies are classified more accurately with fewer terms. Figure 9 shows a comparison of the F-Measure results of the class of reports having an anomaly, when a different number of best terms is picked for each anomaly. The F-Measure is defined as the harmonic mean between precision and recall: $F\text{-}Measure(class) = 2/(1/precision(class) + 1/recall(class))$. The classifier used for this comparison is SVM with a linear kernel and default parameters.

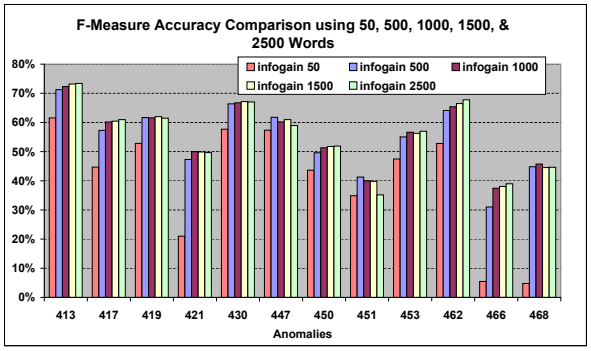

Figure 9. This chart shows the accuracy obtained by using the top 50, 500, 1000, 1500, and 2500 words as measured by information gain in a text classifier for 12 different anomaly codes. Notice that for most anomalies, the accuracy does not significantly increase by using more words.

Experimenting with Different Classifiers

After selecting the optimal number of terms for each anomaly, we tested different classification methods to determine which method would give the best classification accuracy across all anomalies. We experimented with Naive Bayes, Adaboost, SVM, LDA, and Logistic Regression. We use the implementation of SVM in both Weka and SVM-light, and the Weka implementations of Naive Bayes and AdaBoost with base learner Naive Bayes. Figure 10 shows the comparison on the overall precision (both classes) for those methods:

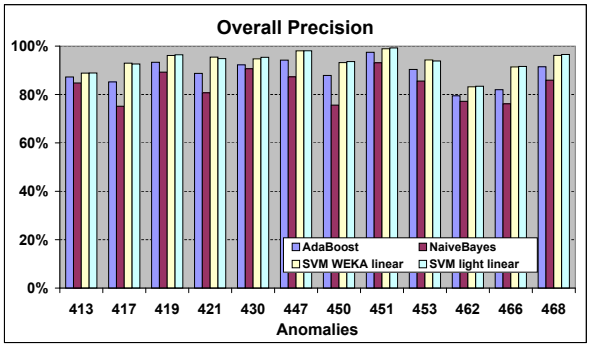

Figure 10. This figure shows a comparison between four different classifiers on 12 anomaly codes. The Support Vector Machine models tend to have the highest precision, with an ensemble method known as AdaBoost close behind.

SVM with a linear kernel performed best overall. Further experiments are mainly performed with the SVM classifier, although we do make comparisons with two other common classification methods - LDA and Logistic Regression. Experimentation with various SVM kernels occurs after normalizing the frequencies of terms remaining after the feature reduction. Unit length normalization did not obtain desirable results. Instead, we normalized by letting f_{ij} be the frequency of term t_i in document d_j. The new frequency f'_{ij} of every term is:

$$f'_{ij} = f_{ij} / \sum_i f_{ij} \tag{29}$$

$$with \sum_i (f'_{ij}) = 1 \tag{30}$$

Our normalization differs from the unit length normalization, which we also tried but did not obtain desirable results. The SVM training and classification are very fast in the SVM-light package. Training and 2-fold cross validation on 20,696 reports takes about 2 minutes on average on a 2 Ghz Pentium III Windows machine with 512MB of RAM.

Support Vector Machines for Text Classification

Support Vector Machines are based on the structural risk minimization principle from statistical learning theory [9]. In their basic form, SVMs learn linear decision rules $h(x) = sign\{\vec{w}\vec{x}\}$ described by a weight vector $\vec{w}$ and a threshold b. Input is a sample on n training examples $S_n = ((\vec{x_1}, \vec{y_1}), ..., (\vec{x_n}, \vec{y_n}))$, $\vec{x_i} \in R^n$,$\vec{y_i} \in \{-1, +1\}$. For a linearly separable S_n, the SVM finds the hyperplane with maximum Euclidean distance δ to the closest training examples. For non-separable training sets, the amount of training error is measured using slack variables ξ_i. Computing the hyperplane is equivalent to solving an optimization problem:

$$minimize : V(\vec{w}, b, \vec{\xi}) = 1/2\vec{w}\vec{w} + C \sum_{i=1}^{n} \xi_i \tag{31}$$

$$subject\ to : \forall_{i=1}^{n} : y_i[\vec{w}\vec{x} + b] \geq 1 - \xi_i \tag{32}$$

$$and : \forall_{i=1}^{n} : \xi_i > 0 \tag{33}$$

The two constraints in Equations (18) and (20) require that all training examples are classified correctly up to some slack ξ_i. If a training example lies on the wrong side of the hyperplane, the corresponding ξ_i is greater or equal to 1. Therefore, $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The parameter C in eqn (17) allows trading off training error and model complexity.

SVMs work well in text classification [10] for a number of reasons:

1. Text normally has high dimensional input space. SVMs can handle large feature spaces.
2. Document vectors are sparse and SVMs are well suited for problems with sparse instances.
3. Most text classification problems are linearly separable. SVMs easily find linear (and for that matter polynomial, RBF, etc.) separators.

SVMs can be implemented with different kernels and for the task of text classification, the most popular are the linear, polynomial and RBF kernels. We experiment with the kernels that we mentioned above and results of the anomalous class F-Measure are shown in Figure 11. As

one can observe, the RBF kernel works best for almost all anomalies. In Figure 12 we show the recall-precision graph for one of the anomalies (anomaly 413). It is evident from the graph that for a relatively low recall we can achieve very high precision.

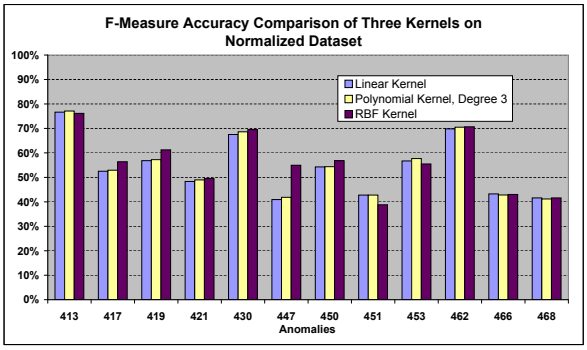

Figure 11. This chart shows the precision for three Support Vector Machine kernels. The RBF kernel tends to have the highest accuracy, although by a small margin.

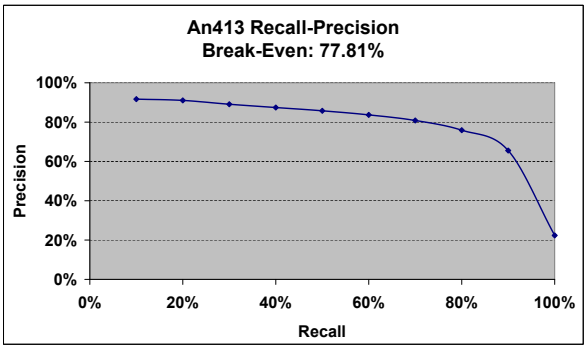

Figure 12. Precision-Recall graph for anomaly 413. This figure shows the classification trade-off between precision and recall. High precision is possible at the expense of a low recall rate.

In order to compare methods, one can compare the break-even point, where precision = recall. If one method has a greater break-even point than another method, then it has higher precision and higher recall. Since we are interested in both high precision and high recall, we will use the break-even point to measure the classification accuracy of multiple classification methods. Results of the break-even point for all anomalies using one method are presented in Figure 13. From those results, we can conclude that for some anomalies we get lower quality predictions than for others. In other words, some anomalies are much harder to classify than others. The problem with the harder to classify anomalies can be related to the initial "bag of words" where the terms picked for those anomalies are apparently not descriptive enough.

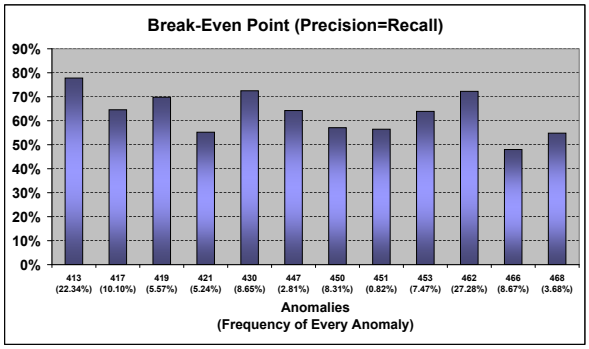

Figure 13. Break-even point for the anomalous class of 12 anomalies for the Support Vector Machine. The frequencies of each anomaly class are shown in parentheses.

Our emphasis is to predict accurately especially on the class that contains a specific anomaly. In other words, we want to be particularly accurate when we predict that an anomaly is present in a report. We call that the anomalous class. Since the frequency of anomalies across reports varies from about 50% to less than 1%, we want to get both high precision and high recall on the anomalous class. That is why we deem using the break-even point of the anomalous class as an evaluation metric to be the most meaningful method of evaluating our results. In Figure 14 we show the break-even comparison of the SVM (RBF kernel) results on the 12 anomalies shown above (Figure 13) with the break-even results obtained from commonly used by statisticians LDA and Logistic Regression classifiers.

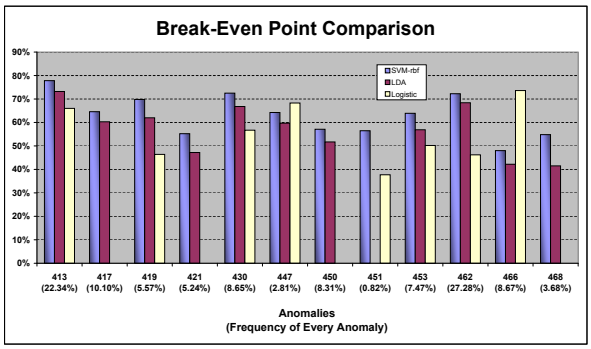

Figure 14. Break-even point for the anomalous class of 12 anomalies, comparison among SVM (blue bars), LDA (red bars), Logistic Regression (yellow bars). The SVM significantly outperforms standard methods.

The results obtained with SVM with an RBF kernel are very good with average anomalous break-even point for all anomalies of 63% and highest of 78%. The non-anomalous average break-point is at the 90%+ level. The break-even results using LDA and Logistic Regression have weighted average anomalous break -even points of 57.26% and 49.78% respectively. Moreover, using Logistic Regression, for 4 of the 12 anomalies, a break-even point could not be produced, the same problem occurred using LDA for 1 of the 12 anomalies. The robust SVM classifier easily produced break-even points for all anomalies. On average, for each of the 12 anomalies it outperforms LDA and Logistic Regression by 5-7% and by 10-15%, respectively.

10. Conclusions and future work

We presented an experimental comparison of the state of the art techniques for text classification applied to the problem of classifying flight reports to predefined categories of occurring anomalies. Starting from the BOW matrix, applying feature reduction techniques and using an SVM classifier, we obtain very good results for some anomalies in terms of both precision and recall. However, for some other anomalies this model does not produce such high levels of desired accuracy. As mentioned above, the problem with the harder to classify anomalies can be related to the initial BOW matrix where the terms picked for those anomalies by the natural language processing methods are not descriptive enough. We plan to investigate the initial reports contents and find NLP methods particularly suited to do better on the currently difficult to classify anomalies. This also points to the importance of well-written reports, particularly when the report describes anomalies which have shown themselves to be difficult to classify.

For recurring anomaly detection on aviation safety reports, finding the semantic relationships between documents is important. For instance, our data sets have quite a few documents with phrases like, ”this problem is similar to another problem.” Approximately 15% of the documents in the CARS data set had similar phrases. Any statistical language model based on the standard ”bag of word” matrix does not embody such semantic information. We call phrases, such as ”similar to” or ”refer to,” trigger phrases. If we could detect the documents which contain trigger phrases and also identify a direct connection to another document, we expect to have a tremendous improvement on the performance of our system. Our results indicate that a large amount of the recurring anomalies which have not been detected by the algorithm are the documents that have trigger phrases. However, the algorithm also found recurring anomalies that the experts had not identified.

To detect the documents which contain trigger phrases and also indicate a connection to other documents, we need to extract the information around the trigger phrase. Information extraction is a well defined research area, and there are many techniques to apply to this problem.

References

[1] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Suvrit Sra, “Generative Model Based Clustering Of Directional Data,” *Conference on Knowledge Discovery in Data, Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[2] Ashok N. Srivastava, Brett Zane-Ulman, “Discovering Recurring Anomalies in Text Reports Regarding Complex Space Systems,” *IEEE*, 2005.

[3] Thorsten Joachims, “Text Categorization with Support Vector Machines: Learning with many relevant features,” Lecture Notes in Computer Science, vol. 1398, *Proceedings of the 10th European Conference on Machine Learning*, April 1998.

[4] Thorsten Joachims, “A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization,” *In Proceedings of International Conference on Machine Learning (ICML)*, 1997.

[5] McLachlan, G. J. and T. Krishnan, “The EM Algorithm and Extensions,” *New York: Wiley*, 1997.

[6] Kanti V. Mardia, Peter E. Jupp, “Directional Statistics,” *Wiley Series in Probability and Statistics*, 2000.

[7] Ian H. Witten and Eibe Frank, “Data Mining: Practical Machine Learning Tools and Techniques,” 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[8] T. Joachims, “Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning,” B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[9] T. Joachims “A Statistical Learning Model for Text Classification for Support Vector Machines,” SIGIR, 2001.

[10] T. Joachims “Text Categorization with Support Vector Machines: Learning with Many Relevant Features,” 1998.

[11] Y. Yang and Jan O. Pedersen “A Comparative Study

on Feature Selection in Text Categorization."

[12] F. Sebastiani "Machine Learning in Automated Text Categorization."

[13] Y. Zhang,J. Callan and T. Minka "Novelty and Redundancy Detection in Adaptive Filtering," *Proceedings of 25th annual ACM SIGIR conference on research and development in information retrieval*, pp. 326–350,2002.

[14] J. Allan, J. Carbonell, G. Doddington,J. Yamron and Y.Yang. "Topic Detection and Tracking Pilot Study: Final Report," *Proceedings of the DARPA Broadcast News Transciprition and Understanding Workshop*, 1998.

[15] Y. Yang ,T. Pierce and J. Carbonell, "A study on retrospective and on-line event detection," *Proceedings of SIGIR'98,*, 1998.

[16] L. Lee , "Measures of Distributional Similarity," *Proceedings of the 37th Annual Meeting of the Association of computational linguistics* College Park, MD, pp. 25-32, 1999.

[17] Y. Zhang, W. Xu , and J. Callan, "Exact Maximum Likelihood Estimation for Word Mixtures," *Text Leaning Workshop at the International Conference on Machine Learing(ICML)*, 2002.