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

Ashok N. Srivastava, NASA Ames Research Center, Ashok.N.Srivastava@nasa.gov,
Ram Akella, 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,
Emmanual D. Pontikakis, University of California, Santa Cruz, manos@stanford.edu
Zuobing Xu, University of California, Santa Cruz, zbxu@soe.ucsc.edu,
Yi Zhang, 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 techniques 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 about 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, the special nature of recurring anomalies (very small cluster sizes) requires incorporating 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. 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 dataset, 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 anomalies by field experts.

Keywords—Target detection, adaptive tests, sequential detection.

Table of Contents

1. Introduction

Aerospace systems have a voluminous amount of information 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 this document database can result in the discovery of valuable information regarding system health monitoring.

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 datasets will also be shown to allow method comparison to be performed by others.

Clustering and classification techniques can be applied to group large amounts of data into known categories. The second problem addressed in this paper is to then autonomously identify recurring anomalies. This approach will be presented and results shown for the CARS dataset. This work has extended uses, including post-analysis for military, factory, automobile and aerospace industries.

2. Brief Look at Classification Methods

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

Non- Parametric Methods

The non-parametric methods in the classification of text documents are generally algorithms like Kmeans and Nearest Neighbor classification. Consider a set of data points distributed in a d dimensional space. Kmeans chooses a set of initial points as the seeds. In step one, each document in the dataset 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 2, the seed associated with each cluster, is updated to the mean of all document vectors in that particular cluster. With the updated seeds, step 1 is repeated again and the process continues iteratively. The documents get assigned to different clusters and the seeds keep getting updated. The algorithm converges when either the seeds stop getting updated or the documents are no longer assigned to different clusters during each iteration. In the following sections we will bring out how this heuristic algorithm is related to the gaussian mixture model.

Parametric Methods

These can loosely be classified as a group of methods that involve parameter estimation. Any mixture model, in particular, a mixture of distributions from the exponential family, can be considered a good example. 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.

Gaussian Mixture Models

The gaussian mixture model assumes that the text documents were generated using a mixture of k gaussian distributions, each with its own parameters θ

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

such that $\sum_i \alpha_i = 1$, where α_i is the prior probability of the ith distribution. Each density 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

The expectation maximization algorithm is an iterative approach to calculate the parameters of the mixture model mentioned above. It consists of two steps: The Expectation step or E-step and Maximization step or the M-step. In the E-step, the likelihood that the documents were generated using each distribution in the mixture model is estimated. The documents are 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 M-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 fit 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 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 ML Estimate of the mean μ_k of the kth distribution is given by,

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

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 assumed values or current estimates of the gaussian parameters μ_k.

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

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 kmeans 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 is a special implementation of the Gaussian Mixture Model, which models the distribution of the underlying data points as a mixture of Gaussian distributions. The parameters are determined by the iterative Expectation Maximization (EM algorithm) of the log likelihood function. The algorithm, however, does not work on sparsely located data points in a high dimensional space.

3. Vector Space Model

The vector space model is a classical way of representing text documents. This representation helps apply machine learning techniques to document classification. 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 the columns are given by the union of all words in all 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 $(\mathrm{i},\mathrm{j})\mathrm{th}$ cell of the BOW matrix corresponds to the TFIDF, which is the Term Frequency Inverse Document Frequency of the $\mathrm{j}\mathrm{th}$ word in the document. The TFIDF is defined as: $TFIDF = TF.IDF$, where $IDF(w_i) = log(n/DF(w_i))$.

Here 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. The BOW matrix is of huge dimension and 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.

4. Directional Statistics

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

Why Use Directional Distribution for Text Data

The preprocessing step before applying the algorithms to text data involves normalization. The TFIDF document vectors are L2 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 if nullified. For Eg: Two documents - one small, one lengthy - on the same topic will have the same direction and hence put in 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 $\mathrm{R}^{\mathrm{d}-1}$ dimensional space. Since it is spherical data, it is more appropriate to use directional distributions.

The von Mises Fisher Distribution

Von Mises Fisher 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.

VMF distribution for a two dimensional circular Random Variable: A circular random variable θ is said to follow a von Mises Distribution if its p.d.f. is given by:

$$\begin{aligned} g(\theta;\mu_o,\kappa) &= \frac{I}{2\phi I_o(\kappa)} \exp \kappa \cos(\theta-\mu_o), \\ & \qquad 0 \le \theta \le 2\phi, \kappa > 0, 0 \le \mu_o \le 2\phi, \end{aligned} \quad (4)$$

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 x is said to have d variate von Mises-Fisher distribution if its pdf is:

$$c_p(k)e^{\kappa\, \mu^T \mathrm{x}} dS^{p-1}, \qquad \mathbf{x} \in \mathbf{S^{p-1}} \subseteq \Re^{\mathbf{p}} \quad (5)$$

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

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

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 hardly a contender for modeling directional data [4].

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 of 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 x_o, if and only if the directional distribution is given by 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 given a fixed circular variance and mean direction μ_o the VMF distribution maximizes the entropy.

Unfortunately there is no distribution for directional data which has all properties analogous to the linear normal distribution. The VMF has some but not all of the desirable properties. The wrapped normal distribution is a strong contender 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 is justified.

5. The VMF Algorithm

In this section we will discuss the theory behind modeling the text documents as a mixture model of VMF distributions. Consider a mixture model consisting of K VMF distributions similar to (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:

$$f(\mathrm{x}/\Theta)=\sum_{\mathrm{k=1}}^{\mathrm{K}}\alpha_{\mathrm{k}}\mathrm{f_k}(\mathrm{x}/\theta_{\mathrm{h}}) \tag{7}$$

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

$$\begin{aligned}E_p[\ln P(\mathrm{X},\mathrm{Z}/\Theta)] &= \sum_{k=1}^{K}\sum_{i=1}^{N}\ln\alpha_k p(\mathrm{x_i}/\Theta)+\\ &\quad\sum_{k=1}^{K}\sum_{i=1}^{N}(\ln f_k(x_i/\theta_k))p(k/\mathrm{x_i},\Theta)\end{aligned} \tag{8}$$

The Maximization Step: In the parameter estimation step or 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:

$$\hat{\alpha}_k=\frac{1}{N}\sum_{i=1}^{N}p(h/\mathrm{x_i},\Theta) \tag{9}$$

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

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

$$\hat{\kappa}_k=\frac{\bar{\mathrm{r}}_k d-\bar{\mathrm{r}}_k^3}{1-\bar{\mathrm{r}}_k^2} \tag{11}$$

where $\mathrm{r_k}=\frac{\mathrm{I}_{d/2}(\kappa)}{\mathrm{I}_{d/2-1}(\kappa)}$

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:

$$p(k/\mathrm{x_i},\Theta)=\frac{\alpha_{\mathrm{k}}\mathrm{f_k}(\mathrm{x_i}/\Theta)}{\sum_{\mathrm{k=1}}^{\mathrm{K}}\alpha_{\mathrm{k}}\mathrm{f_k}(\mathrm{x_i}/\Theta)} \tag{12}$$

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

$$\begin{aligned}q(k/\mathrm{x_i},\Theta)= &\ 1\quad \text{if } k=argmax_{k'}q(k'/\mathrm{x_i},\Theta)\\ &\ 0,\quad \text{otherwise}\end{aligned} \tag{13}$$

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 practise this may be disadvantageous because some data sets like the Reuters data set have multi-labeled documents. Few of the most popular classes in the Reuters dataset 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.

6. Robustness of the Algorithm

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 calculations become untractable because of the huge numbers involved. This gave simulation issues when the algorithms were implemented. So in order to overcome this problem, mathematical approximations were plugged into the update equations. For a modified bessel function of the first kind and order n, for large x, fixed n and x >> n, the approximation is given as follows:

$$I_n(x) \sim \frac{e^x}{\sqrt{2\Pi x}} \qquad (14)$$

7. Datasets Used

We have experimented with several data sets standardly used for text classification.
The 20 News Groups data set: It is a collection of 19997 documents belonging to 20 different news groups. Since the documents in this dataset are primarily email messages, headers such as from, to, subject, organization etc were removed in the preprocessing step. 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. Removing these helps in dimensionality reduction. We were interested only in the body of the messages to keep it a free text classification exercise.

The Diff3 and Sim3 datasets were created from the 20 NewGroups dataset, to verify the performance of the algorithm in well separated classes of documents and documents classes that are closely related to each other in terms of content. Also the size of the dataset has a bearing on the classification accuracy. The more the number of samples to learn the distribution, the better the classification results. So the sim3-small and diff3-small datasets are created with only 100 documents from each class in them.

The CARS Data set: The cars dataset is a collection of problem reports generated by engineers in different fields for the problems in the shuttle. It contains .... documents with a total of .... words in it.

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

The Yahoo!News Groups Dataset: This dataset consists of a collection of 2340 documents belonging to 20 different categories.

8. Simulation Results

Mutual Information: Mutual Information is used as the criteria for comparing the performance of the different methods on the various data sets. Consider two random variables x and y. 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. 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:

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

We used the Mutual Information between the vector of class labels vector produced by the algorithms and the actual class labels of the documents as the criterion to compare the performance of the different algorithms.

To be included: Performance Curves: Comparison of VMF Vs Kmeans: Mutual Information Vs the Number of clusters (averaged over 20 iterations)

- 20 News Groups Diff3 Dataset
- 20 News Groups Sim3 Dataset
- Small Sim3
- Small Diff3
- Yahoo News Groups
- Reuters Dataset

Confusion Matrices to be included Classification confusion matrices for some / all of the above datasets.

Also examples of the top frequency words in each cluster and how they can be representative keywords for the clusters can be included.

9. Text Classification of Flight Reports to Occurring Anomalies

Problem Definition

After each commercial flight in the US, a report is written on that flight describing how the flight went and whether any anomalous events have happened. There is 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 reports in a "bag of words/terms", which contains for all reports their terms, extracted by Natural Language Processing methods, and the corresponding frequencies of the terms. There are a total of 20,696 reports, a total of 28,138 distinct terms, and a total of 62 different anomalies. The anomalies are named with their codes ranging from 413 to 474. 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, we found out that there is not any strong correlation among different anomalies. We concluded that each anomaly has to be treated individually. We, thus, treat the multi-label classification problem as a binary classification problem for every anomaly. As an initial step we pick to work with 12 of the 62 anomalies and try to find a classifier that will perform best for each of them. Our approach can be summarized in three main phases. In the first phase we load the data into a database, collect statistics on it for the purposes of studying the data, then remove the terms with very low frequency. In the second phase we run common feature selection algorithms to reduce the feature space by picking the best terms for every anomaly. In the final phase we experiment with several commonly used for test classification algorithms, such as Support Vector Machines, Naive Bayes, AdaBoost, Linear Discriminant Analysis (LDA), Logistic Regression, implemented in the open-source packages WEKA [1], SVM-light [2] and R. We show convincingly that SVM, with an RBF kernel in particular, performs best for this particular text classification problem. Figure 4.1 summarizes our architecture.

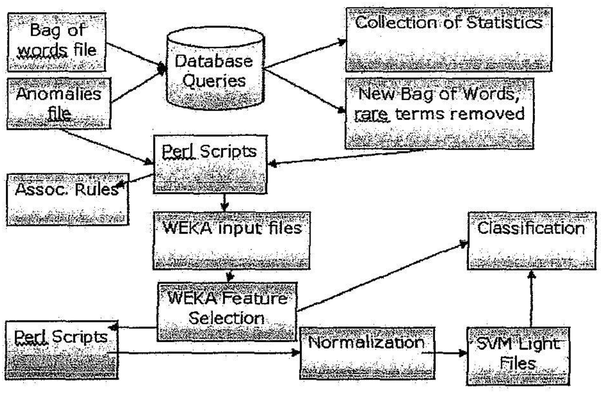

Figure 1. System Architecture
by

Removal of low frequency terms

We remove all terms, regardless of their frequencies, which appear in exactly one report. The intuition behind this is that those terms are not frequent enough to be used for training and will most likely be never 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.

Feature Selection

In this phase we perform feature reduction by selecting the most informative terms for every anomaly [5][6]. We use the Information Gain criterion to rank the terms according to how informative they are for a specific anomaly:

$$IG(class, term) = H(class) - H(class|term) \qquad (16)$$

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. Figure 4.2 shows comparison of the F-Measure (the harmonic mean between precision and recall) results of the class of reports having an anomaly, when different number of best terms is picked for each anomaly. The classifier used for that comparison is SVM with a linear kernel and default parameters.

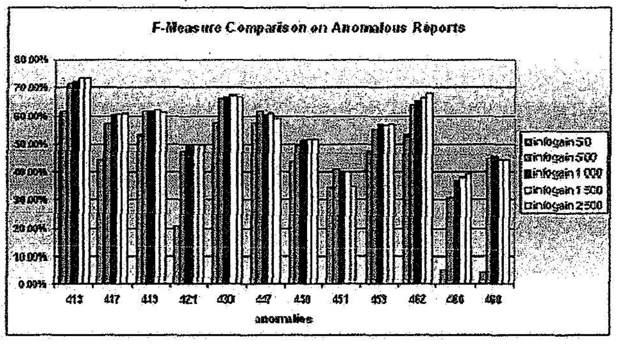

Figure 2. Figure 4.2. Number of terms, comparison
by

An observation is that anomalies that are not occurring so frequently are classified more accurately with less number of terms. This seems rather reasonable since it makes sense that less frequently occurring anomalies would be described well enough with just a few terms.

Experimenting with different classifiers

After we select the optimal number of terms for each anomaly, we test different methods for classification. We experiment with Naive Bayes, Adaboost, SVM, LDA, Logistic Regression. At that point we want to find which method would give the best classification accuracy across all anomalies. The histogram in Figure 4.3 shows the comparison on the Overall Precision (both classes) for those methods:

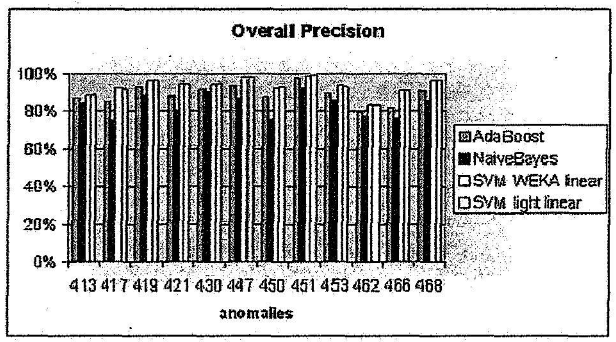

Figure 3. Figure 4.3. Classifiers comparison
by

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. SVM with a linear kernel performs best on all anomalies. We, therefore choose to experiment further mainly with the SVM classifier, although later we do make comparisons with two other common classification methods - LDA and Logistic Regression.

Support Vector Machines for text classification

Support Vector Machines are based on the structural risk minimization principle from statistical learning theory [3]. 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:

$$minimize : V(\vec{w}, b, \vec{\xi}) = 1/2\vec{w}\vec{w} + C\sum_{i=1}^{n} \xi_i \tag{17}$$

$$subject\ to : \forall_{i=1}^{n} : y_i[\vec{w}\vec{x} + b] \geq 1 - \xi_i \tag{18}$$

$$and : \forall_{i=1}^{n} : \xi_i > 0 \tag{19}$$

The constraints (2) 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 (1) allows trading off training error and model complexity.

SVMs work well in text classification [4] for a number of reasons:

1. Text normally has high dimensional input space. SVMs use overfitting protection which does not depend on the number of features and therefore have the potential to 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 most popular are the linear, polynomial and RBF kernels. We experiment with all those kernels after we normalized the frequencies of terms remaining after the feature reduction. Let f_{ij} be the frequency of term t_i in document d_j. Then based on our normalization, the new frequency f'_{ij} of every term is:

$$f'_{ij} = f_{ij} / \sum_i f_{ij} \tag{20}$$

$$with \sum_i (f'_{ij}) = 1 \tag{21}$$

Our normalization differs from the unit length normalization, which we also tried but did not obtain desirable results. We experiment with the kernels that we mentioned above and results of the anomalous class F-Measure are shown in Figure 4.4. As one can observe, RBF kernel works best for almost all anomalies. In Figure 4.5 we show the recall-precision graph for one of the anomalies (code 413). It is evident from the graph that for a relatively low recall we can achieve very high precision.

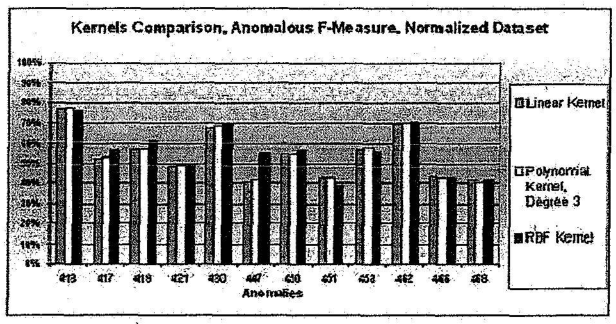

Figure 4. Figure 4.4. Kernels comparison
by

Results of the break-even point (precision = recall) for all anomalies are presented in Figure 4.6. From those results, we can conclude that for some anomalies we get lower quality predictions than for others. In other words, some anomalies

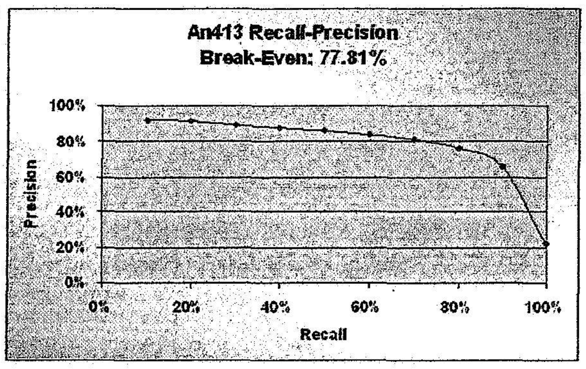

Figure 5. Figure 4.5. Recall-Precision graph for anomaly 413

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.

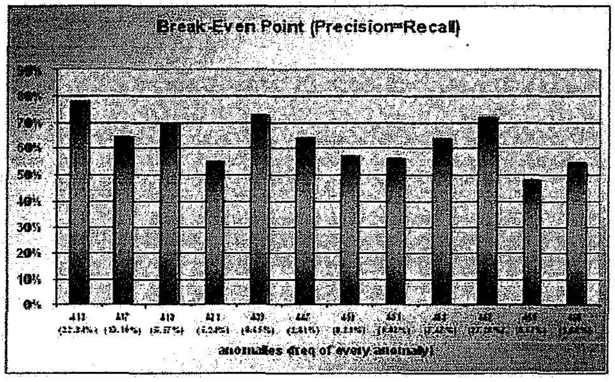

Figure 6. Figure 4.6. Break-even point for the anomalous class of 12 anomalies, SVM

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.

SVM results comparisons with LDA and Logistic Regression results

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 4.7 we show the break-even comparison of the SVM (RBF kernel) results on the 12 anomalies shown above (Figure 4.6) with the break-even results obtained from commonly used by statisticians LDA and Logistic Regression classifiers.

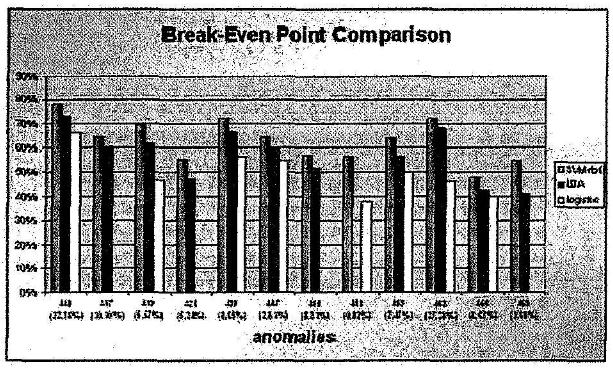

Figure 7. Figure 4.7. Break-even point for the anomalous class of 12 anomalies, comparison among SVM, LDA, Logistic

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 have weighted average anomalous break - even points of 57.26% and 49.78% respectively. Moreover, using Logistic, on 4 of the 12 anomalies, a break-even point could not be produced, and using LDA on 1 of the 12 anomalies. The robust SVM classifier easily produces break-even points for all anomalies. On each of the 12 anomalies it outperforms LDA by 5%-7% on average and Logistic by 10%-15% on average.

10. Recurring Anomaly Detection

The Recurring anomaly detection problem that we address in this paper is as follows. Given a set of N documents, where each document is a free text English document that describes a problem, an observation, a treatment, a study, or some other aspect of the vehicle, automatically identify a set of potential recurring anomalies in the reports. Note that for many applications, The corpus is too large for a single person to read,understand, and analyze by hand. Thus, while engineers and technicians can and do read and analyze all documents that are relevant to their specific subsystem, it is possible that other documents, which are not directly related to their subsystem still discuss problems in the subsystem. While these issues could be addressed to some degree with the addition of structured data, it is unlikely that all such relationships would be captured in the structured data. Therefore, we need to develop methods to uncover recurring anomalies that may be buried in these large text stories. Overall recurring anomaly detection helps to identify system weakness and avoid high-risk issues. The discovery of recurring anomalies is a key goal in building safe, reliable, and cost-effective aerospace systems. Furthermore, recurring anomaly detection can be applied to other domain , such as computer network security and health care management.

From the research perspective, recurring anomaly detection is 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. [7]. A novelty and redundancy detection distinguishes among relevant documents that contain new (novel) information and relevant documents that do not . The definition of recurring 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 retrospective event detection task in Topic Detection and Tracking [8] [9]. The retrospective detection task is defined to be the task of identifying all of the events in a corpus of story. Recurring anomaly detection task differs from their task in having many single document clusters. However , the similarity of the tasks are worth exploring, and several methods we investigated are motivated by their work. The core part of our work is the similarity measures between statistical distributions. There has been much work on similarity measures. A complete study on distributional similarity measures is presented by [10].

Language Models and Similarity Measures

There are two general approaches to measure the similarity between documents: non statistical method and statistical method. 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 $d = (w_1(d), w_2(d), \ldots, w_n(d))^T$, then:

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

The statistical method is to measure the similarity between different distributions. Each distribution generates one document, while in the generative model we used in the previous section each distribution generates a cluster of documents. In our recurring anomaly detection problem, 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 (VMF) distribution, which we used in the previous section, is that we can not estimate the mean and the variance unless we have certain amount of data in each cluster. To estimate the parameters of VMF distribution with single sample returns the mean as the document vector itself and zero variance.

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

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

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

Clearly, now the problem essentially reduced to a multinomial distribution parameter estimation problem. The maximum likelihood estimation of the probability of a word occurring in the document is

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

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 user-specific Topic model , and a document-specific information model. Each word is generated by each of the three language models with probability λ_E,λ_T and λ_{dcore} respectively:

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

where $\lambda_E + \lambda_T + \lambda_{dcore} = 1$.

For instance, in a short document "the airplane engine has some electric problems.", the words "the" , "is" 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 new information model . Because all the documents are anomaly reports on airplane, the 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 problem. So only measuring the similarity between the document specific models makes the recurring anomaly detection more accurate.

If we fix λ_E,λ_T and λ_{d_core}, then there exists a unique optimal value for the document core model that maximizes the likelihood of the document.

We employ quick algorithm based on Lagrange multiplier method to find the exact optimal solution, given fixed mixture weights [11].

We need some metrics to measure the similarity between multinomial distributions. Kullback-Leibler divergence, a distributional similarity measure, is one way to measure the similarity of one multinomial distribution given another.

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

The problem with KL divergence is that if a word never occurs in document, it will get a zero probability $p(w_i|d) = 0$. Thus a word in 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, Joccard's Coefficient and skew divergence. Jensen-Shanon divergence [10] has been proved to be a useful symmetric measure of the distance between distributions

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

We also employ skew divergence [10] to measure the similarity between two discrete distributions. Skew divergence is an asymmetric generalization of the KL divergence,

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

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 Jesen-Shannon divergence. In our experiment, we choose $\alpha = 0.99$ to approximate the KL divergence and avoid singularity.

The Joccard's coefficient differs from all the other measures. We consider in that it is essentially combinatorial, being based only on the sizes of the supports of document specific distribution rather than the actual value of the distribution

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

Based on the similarity measurement between anomaly documents, we apply agglomerative hierarchical clustering method to partion the documents. The aggolomerative hierachial algorithm produces a binary tree of clusters in a bottom-up fashion: the leaf nodes tree are single document clusters; a middle-level node is the centriod 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 aggolomerative hierarchial clustering method we appy is single linkage clustering. The defining feature of the method is that similarity between groups is defined as the similarity between the closest pair of objects, where only pairs consisting of one object from each group are considered. We set up a threshold on the similarity to obtain the parition which yielded the optimal result.

New Performance Measures for Recurring Anomalies

The recurring anomaly detection problem can be decomposed into two parts: detecting 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 1, we see that our algorithm groups the documents into 4 clusters. The column "Expert" shows the expert clustering results.

Table 1. 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	

In this example the algorithm has made the following mistakes: missing recurring anomaly 8; detecting non recurring anomalies 5 and 6; separating recurring anomalies 1,2,3,4 into two clusters; separating recurring anomalies 9,10 into two clusters and combining recurring anomalies 1,2,5 into one cluster. So we summarize the mistakes into four categories: 1.missing recurring anomaly, 2.detecting non recurring anomaly. 3.separating same kind of recurring anomalies into different clusters. 4.combining different kinds of recurring anomalies into one cluster. The standard precision and recall measure can only characterize the first two mistakes, so we need to devise another metric to measure the last two mistakes. In our problem,

$$\text{Precision} = \frac{R^+}{R^+ + N^+}$$

$$\text{Recall} = \frac{R^+}{R^+ + R^-}$$

R^+ R^- N^+ and N^-correspond to the number of documents that fall into the following categories

Table 2.

	Labeled by Expert	Not Labeled by Expert
Detected	R^+	N^+
Not detected	R^-	N^-

The number of anomalies which are both detected by algorithm and labeled by expert is 6. The number of anomalies detected by algorithm is 9, and the number of anomalies labeled by expert is 8. So the precision is 0.67 and the recall is 0.75.

Precision and recall measure the accuracy of detecting recurring anomalies, but do not characterize the accuracy of clustering anomalies. Because the anomalies, which have not been either detected by algorithm and or labeled by expert, do not affect the accuracy of the clustering, we delete these anomalies. The remaining anomalies are shown in table2.

Table 3. Simple clustering example for illustrating new performance measure (after deleting the documents which are not deteced both by algorithm and experts)

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

To measure the mistakes that caused by separating same kind of recurring anomalies into different clusters, we add up the reciprocal of the number of splited clusters and normalized by the total number of clusters in expert result. If the algorithm result exactly match the expert result, we get score 1. The score decreases as the number of splited cluster increases. The other point view of the miscombination by algorithm is misseparation by expert. So we use the same scheme but based on algorithm result to calculate miscombination score. The method to score the misseparation and miscombination is defined as following,

$$\text{Misseparation} = \frac{\sum_{\text{expert cluster}} \frac{1}{NSA}}{NE}$$

$$\text{Miscombination} = \frac{\sum_{\text{algorithm cluster}} \frac{1}{NSE}}{NA}$$

where

NSA	=	number of expert clusters which contain the anomalies in each algorithm cluster
NSE	=	number of expert clusters which contain the anomalies in each algorithm cluster
NE	=	number of clusters in expert result
NA	=	number of clusters in algorithm result

It's better to understand the measure scheme by explaining it with the example. The algorithm separates anomaly 1, 2, 3 and 4 in expert cluster 1 into 2 clusters, so the misseparation score for this cluster is 1/2; the misseparation score for expert cluster 2 is 1; and the score for cluster 3 is 1/2. The overall score for separation is 1/2+1+1/2=2. To normalize the score, we divide it by the number of clusters in the expert result. So the normalized misseparation score is 0.75. The miscombination score is calculated in the inverse direction.

Experimental Results

The aerospace system we analyzed is the NASA Space Shuttle problem reports as represented in the CARS dataset, which consists of 7440 NASA Shuttle problem reports. These reports come from the three subsystems.

Some domain experts read the anomaly reports and provide a clustering results. According to their results , among total 7440 reports, there are 1553 recurring anomalies, which are grouped into 366 clusters. Consequently, there are 7440 − 1553 = 5887 single document clusters, which make this problem distinct.

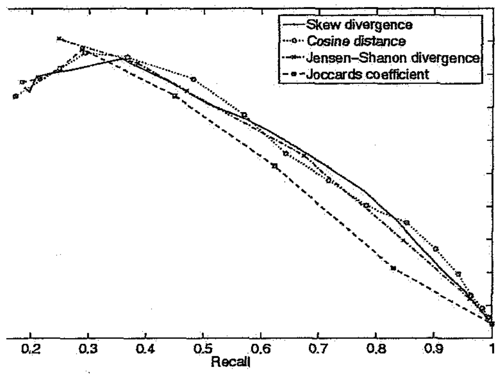

Figure 8. Comparing Precision and Recall Measure on CARS Data

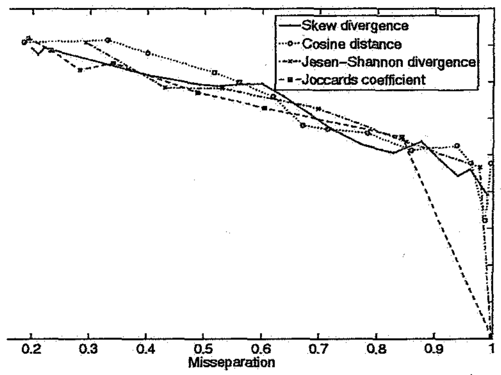

Figure 9. Comparing Misseparation and Miscombination Measure on CARS Data

Four similarity measures :cosine distance, skew divergence, jenson-shanon divergence and joccard's coefficient are compared on the CARS data set. Figure 8 and Figure 9 summarize the effectiveness of four similarity measure schemes.

The skew divergence based on word mixture model and the cosine distance are very effective. In general, they outperforms all the other methods. The Joccard's coefficient measure is the least accurate. It is very suprise that the traditional cosine similarity metric is very effective, because cosine similarity is less well-justified theoretically than the 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 recall ranges from 0.55 to 0.85, the skew divergence is most accurate. This region satisfies the user requirements: relatively high recall and low precision.

To testify the effectiveness of the word mixture model, we compared the performance of skew divergence measure based on mixture model and general language model. The results are shown in Figure 10 and Figure 11. We see that the mixture model result is consistently more accurate than the general model.

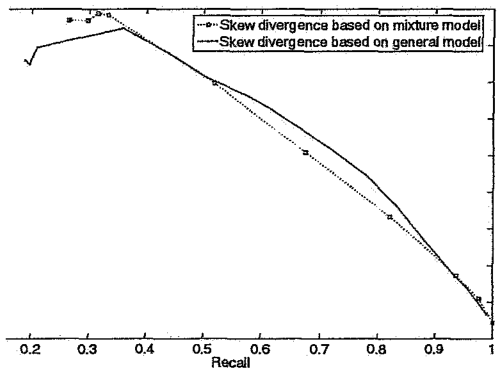

Figure 10. Comparing Precision and Recall Measure for Mixture Model

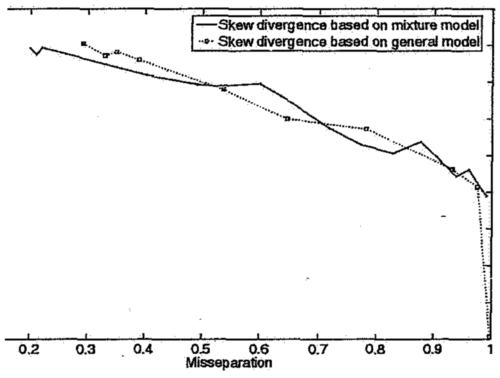

Figure 11. Comparing Misseparation and Miscombination for Mixture Model

11. CONCLUSIONS AND FUTURE WORK

Difficult to Classify Anomalies:

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 "bag of word", 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 "bag of words" 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 suited particularly to do better on the currently harder to classify anomalies. We can also address the problem by making suggestions at the base level of how the reports themselves should be written, particularly when describing events such as those anomalies which are difficult to classify at the present time with the currently given "bag of words".

Future direction: Semantics or Statistics?

Semantics or statistics? This is a question which has puzzled everyone working in text mining field. For Recurring anomaly detection on airplane problem reports , finding the semantics between documents is much more important than devising a good statistical language model. Because our data set has quite a few documents, which is written in a way such as " this problem is similar to another problem". Any statistical language model based on bag of word matrix does not embody such information.

We call the word "similar to" "refer to " as trigger word. If we could detect the documents which contain trigger word and also indicate a connection to other documents, we will have a tremendous improvement on the performance of our system. We checked the results and found that a large amount of the recurring anomalies which have not been detected by the algorithm are the documents that have trigger words. However, the algorithm also found quite a few recurring anomalies that the experts has not found, so we sent our results to the experts to reevaluate.

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

REFERENCES

[1] Ian H. Witten and Eibe Frank (2005) "Data Mining: Practical Machine Learning Tools and Techniques", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[2] T. Joachims, "Making large-Scale SVM Learning Prac-

tical. Advances in Kernel Methods - Support Vector Learning", B. Scholkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[3] T. Joackims "A Statistical Learning Model for Text Classification for Support Vector Machines", SIGIR, 2001.

[4] T. Joackims "Text Categorization with Support Vector Machines: Learning with Many Relevant Features",1998.

[5] Y. Yang and Jan O. Pedersen "A Comparative Study on Feature Selection in Text Categorization".

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.