Text Classification with Evolving Label-sets

Shantanu Godbole
IIT Bombay
shantanu@it.iitb.ac.in

Ganesh Ramakrishnan
IBM India Research Lab
ganramkr@in.ibm.com

Sunita Sarawagi
IIT Bombay
sunita@it.iitb.ac.in

Abstract

In this paper, we introduce the evolving label-set problem encountered in building real-world text classification systems. This problem arises when a text classification system trained on a label-set encounters documents of unseen classes at deployment time. We design a Class-Detector module attached to text classification systems that monitors unlabeled data, detects new classes, and suggests them to the human administrator for deployment in the original label-set.

A central notion in our algorithms is the use of abstractions that group together tokens under human understandable concepts and also provide a mechanism of assigning importance to unseen terms. We present algorithms for selecting documents for a new class based on state-of-the-art generative models and high performance discriminative classifiers. Experiments on three real world taxonomies show that the generative models can select documents comprising a new class with high precision, and also automatically trigger the emergence of new classes with more than 85% precision.

1. Introduction

Numerous applications like spam filtering, e-mail routing, Web directory maintenance, and news filtering, have fueled extensive text classification research in recent years [10, 12, 16]. State-of-the-art classifiers routinely achieve up to 90% accuracy on well-known benchmarks, and this is surpassed in highly tuned industry-grade text classification systems [1]. Most text classification research assumes a fixed representation of features like bag-of-words, and a partially labeled corpus. Statistical learners also depend on the deployment to be reasonably related to the training. Not all these assumptions hold in real-life.

One important challenge in building text classification systems is that the constitution of unlabeled data changes over time. Often new classes are introduced and need to be detected and folded into the system. We call this the **evolving label-set** problem. For example, consider a classification problem with n classes, where the classes are documents about certain countries (*India*, *US*, *UK*, . . .). Over a period of time, a new country's documents (say *Australia*) are introduced into the system. The evolving label-set problem is to detect such (one or more) new classes, propose a cohesive set of documents for training the new classes, get user for validation about these fitting in with the label-set, and fold these new classes into the classification system. Such problems occur especially when a nascent classification system is built from scratch, the entire set of labels is not known beforehand, and the user's understanding of the label-set evolves over time.

Such phenomenon is common place in directory systems like Dmoz[1] that manually classify ever-changing webpages. For example, a directory of scientific disciplines would need to add "bio-informatics" as it emerged as a new discipline, or add an industry type "cell-phones" when they started becoming popular. Another example is in the news domain where new kinds of news stories about recent events need to be detected and classified. This example is well studied and called *novelty detection* in the topic detection and tracking (TDT) track at the TREC [2] conferences. The novelty detection task aims for online clustering of news stories; it's goal is to tag novel news stories as interesting and spawn new events for these stories. The evolving label-set problem on the other hand, occurs in a classification setting and new classes need to be carefully spawned only if they fit into the existing label-set. We review some of the novelty detection work and point out differences to the evolving label-set problem in Section 6.

Our main contributions in this paper are as follows: We design algorithms for identifying new classes in both discriminative and generative settings. We introduce the notion of abstractions so as to capture the importance of terms not encountered during training, and also to provide a representation that more intuitively reveals the classification criteria to the user. We perform experiments on three very different types of real-life taxonomies and show that our methods

[1] http://dmoz.org
[2] http://trec.nist.gov

achieve 60–90% precision of discovering unlabeled documents comprising a new class. We also make the surprising discovery that while discriminative methods are more accurate than generative ones, as far as the discovery of new classes is concerned they fare poorly with respect to state-of-the-art generative methods.

2. Problem Setting

We envision a scenario where a separate module for new class detection continuously monitors unlabeled documents as they arrive into a text classification system for label assignment. When the module, that we call Class-Detector, gathers enough evidence of an emerging new class, it sends a trigger to the administrating user. Alternately, the user could periodically query the Class-Detector for the presence of a new class. The Class-Detector then presents to the user a ranked list of documents that could comprise a new class. The user can then choose to add a new class to the classification system with an initial labeled set filtered from this ranked list.

Our methods for tackling the evolving label-set problem *do not interfere* with the working of the main classifier. The main classifier can be any high-performance well-tuned algorithm; a popular choice being Support Vector Machines (SVMs) [10]. As we will see later in Section 5 the underlying learning model that works best for the detecting new classes can be very different from the optimal main classifier for label assignment.

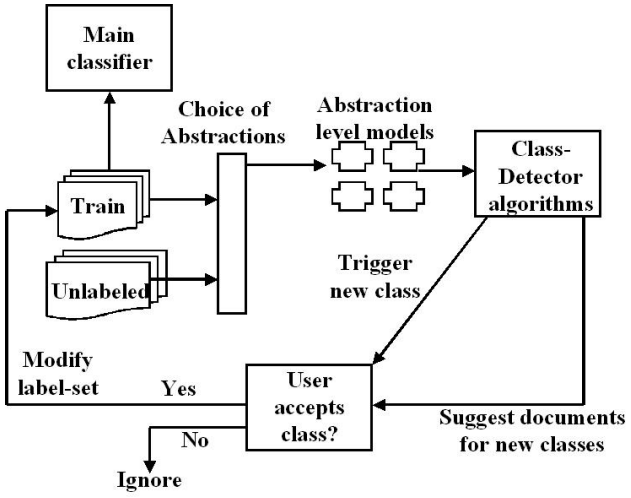

Figure 1. System overview

The two main technical components of the Class-Detector are:

1. The **new class trigger** that decides if there is enough consistent divergence in the unlabeled set to define a new class, and,
2. The **document selector** that picks a ranked list of documents comprising a possible new class.

Intuitively, we expect to find a new class when there are a significant number of unlabeled documents that do not fit the existing class structure and which are themselves coherent enough to be grouped into a class. Converting this intuition into a robust procedure poses several challenges.

Firstly, separating out the documents forming a possible new class from mis-classified documents of existing classes on the basis of being "misfits" in the classification model, is extremely challenging, particularly in the presence of multi-labeled documents. If the selected documents contains several of these mis-classified documents (say more than 50%), the user may get confused about the nature and scope of the proposed new class. In fact, for state-of-the-art one-vs-others SVM ensembles, we have observed that as many as 30% of the unlabeled documents are rejected by all the binary classifiers, making them hard to separate from valid new class documents. We therefore also explore state-of-the-art generative models to capture the degree of fit of unlabeled documents.

Secondly, it is likely that documents of new classes will contain terms that have not been seen during training, and not all of these are important for classification. Further, even normal unlabeled documents contain new terms in abundance, thereby eliminating the possibility of depending on frequency of new terms to detect new classes. In a supervised setting, the importance of terms is established either explicitly using statistical metrics like information gain, or implicitly, in the classifier via term weights (as in SVMs). Such metrics that depend on labeled data are not applicable here. We depend on indirect methods to establish term importance via a notion of term **abstractions** that assigns importance to a family of terms together. Abstractions indicate various properties of the term based on the way it is used in the documents. Examples of abstractions are: Named-Entity (NE) tags, part-of-speech (POS) tags, formatting features, visual clues in HTML documents, and match with external dictionaries or keyword ontologies. For example, a classification system based on regions would assign high importance to location NE tags. Additionally, abstractions help a user better interpret the criteria used for defining new classes. Abstractions can be clubbed together to form *abstraction sets*. The user can choose a small number of candidate abstraction sets for a label-set by inspection, domain knowledge, or through validation experiments in an off-line training phase.

Abstractions – an example: We highlight the importance of abstractions in understanding a label-set in Figure 2. With the full vocabulary, for three classes, we show the four most indicative features used in classification. We also

show indicative features when only the organization name NE tag is used for representing documents.

Full vocab
bank,issue,warrant,fee oil,crude,million,refinery compuserve,service,subscribe,cost
Organization names
Commonwealth Bank of Aus, Commerzbank, Central Bank, Fleet Fin Group Gulf Oil, Chinese Petro Corp, Esso Aus Ltd, Natural Gas Corp Europe Online, Compuserve, First Data Corp, AOL

Figure 2. Indicative features for a label-set

The full vocabulary makes it hard for us to judge what the classes in the label-set are though we can estimate that the classes are broadly about commerce, oil, and computers. However, looking at only the organization names (the organization name abstraction), we immediately understand that the label-set is about industry types and the classes pertain to a kind of banking, oil companies, and computer data companies respectively. Indeed, this label-set is taken from the *Industries* dataset described in Section 5. If new classes are discovered in unlabeled data based on the full vocabulary, it is not clear that the user will be able to judge the nature or constitution of the proposed class. On the other hand, the organization name abstraction will definitely help the user in understanding and identifying a new industry type.

The rest of the paper is organized as follows. In Section 3 we present algorithms for selecting a ranked list of documents comprising a possible new class in a generative and discriminative setting. In Section 4 we present our method for automatically triggering the presence of a new class. Finally, in Section 5 we present detailed experimental results on real-life datasets.

3. Selecting documents for a new class

In this section we propose generative and discriminative methods for selecting unlabeled documents for a likely new class. First, we present an algorithm for generative classifiers based on the notion of *support*. Following this, we present algorithms for discriminative classifiers based on the notion of classification *confidence*.

3.1. Generative methods

Generative models for text such as LDA [5], Aspect [9] and BayesANIL [14] model the process of generation of documents and document features (*e.g.* words) from classes. BayesANIL is a Bayesian model that assumes conditional independence of words (features) from classes, given documents, and has been shown to be capable of estimating uncertainties associated with the labeling process. Given a corpus of documents $d \in \mathcal{D}$ and some of the documents labeled with class labels $c \in \mathcal{C}$, the model estimates the joint distribution $Pr(c, d)$ of training documents d and class labels c by using a generalization of the Expectation Maximization (EM) algorithm. The $Pr(c, d)$ estimate from BayesANIL can be interpreted as a measure of *support* for membership of document d in class c. The marginalized probability $Pr(d) = \sum_{c \in \mathcal{C}} Pr(c, d)$ can be interpreted as a measure of *support* of how well document d fits into the existing label-set defined by classes $\mathcal{C} = \{c_1, c_2 ... c_n\}$. We chose BayesANIL over other generative models, because in empirical experiments reported in [14], the probability estimates $Pr(c, d)$ from BayesANIL reflected the notion of support in the contexts of text classification in the presence of noisy, approximate and incomplete labeling. Additionally, BayesANIL provides for folding in feature evidence from unlabeled documents, which is especially important, given that some features are often poorly represented in the labeled set. This folding-in is enabled by setting the parameter λ in BayesANIL to a non-zero value. In our experiments, we used $\lambda = 0.001$.

Thus, the parameters $Pr(c, d)$ and $Pr(d)$, provide for documents from the model, a good notion of *support*. We use this measure of *support* for the problem of detecting documents pertaining to classes beyond those already provided; documents with low support for membership in any of the existing classes are determined as documents of a candidate new class. In general, we could make use of any generative model that (1) provides an estimate of the joint distribution $Pr(c, d)$ or the *support* for membership of each document in each class and (2) provides for folding in feature evidence from unlabeled documents.

We now discuss algorithms for selecting documents from the unlabeled set to form a candidate new class. A simple method of selecting documents belonging to a new class is to select documents with the lowest $\Pr(d)$ values. We call this method *SortPrD*. Another method for suggesting new class documents is to seed an $(n+1)^{th}$ class by documents with the lowest $Pr(d)$ values, re-train the generative model for $(n+1)$ classes, and select unlabeled documents with the highest $Pr(c_{n+1}, d)$. We call this method *PrDNewClass*. Since sorting based on $Pr(d)$ is not perfect, it is likely that documents selected by both these methods will include documents of existing classes that were mis-classified either due to noise or for being multi-labeled. Next, we propose an algorithm that avoids this limitation.

The *GenSupp* algorithm: We project all training and unlabeled documents in an n-dimensional *support space* where the components for a document d along the n dimensions are its n $\Pr(c, d)$ values. For training documents these $\Pr(c, d)$ values could be pre-computed once during

the training phase and stored. We then use a hierarchical clustering (HAC) algorithm to group similar documents. We measure the distance between any two documents d and d' by the average KL-distance between their $\Pr(c, d)$ and $\Pr(c, d')$ n-dimensional scores. If d_is and d'_is are the document projections in the support space, then distance between d and d' is given by:

$$dist(d, d') = \frac{\sum_{k=1}^{n} d_k \ln(\frac{d_k}{d'_k}) + \sum_{k=1}^{n} d'_k \ln(\frac{d'_k}{d_k})}{2}$$

We tried single-link, complete-link, group-average, and Ward's method as cluster combination strategies, and found Ward's method [6] to work best. We grew the dendrogram till we had a large number (say $5n$) of small clusters. Since $Pr(d)$ scores gives the probability of generating the document from the model, we expect the lowest $Pr(d)$ values to be assigned to the new class or other noisy, multi-labeled documents. We found this to be true empirically. To get tight sub-clusters from these candidates, we chose clusters which had the lowest average value of $Pr(d)$ of its constituents. Figure 3 gives the complete *GenSupp* algorithm (for Generative model method based on Support). We require each candidate cluster to have a minimum number of unlabeled documents (say 5) to guard against outliers and to be able to define a new class. We also require clusters to be *pure* where the fraction of unlabeled documents is at least $p\%$; this ensures that the new class lies in an area of the support space where there are no (or few) training documents in the vicinity. We chose p as 20% in our experiments.

1: $CandidateClusterSet = \{\phi\}$
2: $PureClusterSet = \{\phi\}$
3: Project all training and unlabeled docs in n-dimensional space on $Pr(c_i, d)$ scores
4: Perform HAC using Ward's method and average KL-distance
5: Grow the dendrogram to $5n$ clusters
6: **for all** Clusters **do**
7: If cluster has a minimum threshold number of unlabeled documents: add it to $CandidateClusterSet$.
8: **end for**
9: **for all** Clusters in $CandidateClusterSet$ **do**
10: If cluster has a minimum fraction of unlabeled documents vs. labeled: add it $toPureClusterSet$
11: **end for**
12: Select the cluster from $PureClusterSet$ with lowest average value of $Pr(d)$ of its constituents
13: Sort unlabeled documents in this cluster by $Pr(d)$

Figure 3. *GenSupp* **algorithm**

3.2. Discriminative methods

Discriminative classifiers like SVMs do not model a document's generation probability, therefore we do not have equivalents of $\Pr(d)$ values for selecting candidate documents for a new class. We assume a standard one-vs-other binary ensemble for multi-class classification and design two algorithms that rely on the "rejection" scores of the binary classifiers of each of the n classes.

NOTA-based method – *NotaSVM*: Let NOTA denote the set of unlabeled documents that are rejected by all the binary SVMs (NOTA stands for None Of The Above classes). Some of these documents are possibly self-similar, coherent and belong to a candidate new class; others may be off-topic, noisy, or multi-labeled. Un-tuned SVMs are known to produce a significant fraction (up to 30%) of such NOTA predictions. We train a $(n+1)^{th}$ binary SVM with the NOTA set as the positive class and the known training data as the negative class. We expect this SVM to prefer documents of the new class and accordingly select candidate new class documents in decreasing order of their scores from the $(n+1)^{th}$ binary SVM. We call this algorithm *NotaSVM* and note that it is similar in spirit to *PrDNewClass*.

Require: n-class SVM one-vs-others ensemble
Require: n-class outputs for training and unlabeled data
1: Seed a $(n+1)^{th}$ class with NOTA documents returned by the n-class ensemble
2: Re-train new $(n+1)$ class SVM one-vs-others ensemble and apply it to unlabeled documents
3: Rank documents classified positively by the $(n+1)^{th}$ binary SVM by distance from separator

Figure 4. *NotaSVM* **algorithm**

We propose a second algorithm along the lines of the *GenSupp* algorithm for generative models.

***DisConf* algorithm:** The *DisConf* algorithm is designed to be the HAC-based discriminative counter-part of the *GenSupp* algorithm. In *GenSupp* we represented documents in the $Pr(c, d)$ space. In this case, we represent each document by the n-dimensional vector of projection scores from the SVM ensemble. These scores indicate the prediction *confidence* of each classifier. The distance between two documents is the Euclidean L_2 distance metric. An important difference between *DisConf* and *GenSupp* is that in *DisConf* we cannot choose a tight cohesive cluster based on the lowest average value of a ranking function like $Pr(d)$. Instead, we need to apply a heuristic like choosing a cluster which has the most negative average value of document projections. Remember that for a document d, most of the prediction scores in its confidence vector will be negative.

As we will see in the experiments in Section 5, this is not a very good heuristic for choosing candidate new classes. The algorithm is same as 3, except for two differences (1) each document is projected in the n-dimensional space of SVM ensembles output confidence scores instead of the $Pr(c_i, d)$ scores and (2) the L_2 distance metric is used instead of KL-distance.

4. Automatically triggering new classes

We now consider the problem of detecting if the selected documents indeed comprise a new class or not. In general, for a classifier there will be several unlabeled documents not classified in any of the existing classes (NOTA documents in case of SVMs) or having very low probability of being generated by the learned model (low $Pr(d_j)$ in BayesANIL). Typically, most of them are due to noisy and mis-classified multi-labeled documents and automatically triggering if there is a new class amongst them is a hard problem.

We approach the problem using BayesANIL's notion of support using its $Pr(c, d)$ scores as follows. Let $T = \{T_1, T_2, \ldots, T_n\}$ be the training documents for the original n-class label-set. We keep aside a set $V = \{V_1, V_2, \ldots, V_n\}$ of documents for measurement. Another set $U = \{U_1, U_2, \ldots, U_n\}$ are treated as unlabeled documents. We introduce a fake class and change the original label-set by adding T_{n+1} which is a cohesive set of documents in U_i found by HAC as in *GenSupp*. T_{n+1} is a fake class as it's documents are chosen from some U_i; for every such U_i, it's corresponding class T_i already exists in the original label-set. We re-train this $n + 1$ class document collection using BayesANIL and find the value of two measures MV and MT:

$MV = \sum_{d \in V} Pr(c_{n+1}, d)$, and

$MT = \sum_{d \in T_{n+1}} Pr(c_{n+1}, d)$.

MV measures the support for the validation set V from the newly added class, and MT measures the support of the newly added class for itself. We perform this experiment n times, every time adding documents of an existing class as a fake class. This gives us n prototype values which we store as MV_i and MT_i, where $i = 1 \ldots n$. These MV and MT vectors prototype the range of values these support measures take when fake classes are introduced into the label-set.

We expect that if we really detect a new class from the unlabeled data, then it's corresponding MT_{n+1} value should be higher that all previous MT_i's. Since the fake classes always had a corresponding class in T, these documents in T_{n+1} share the probability mass of $d \in T_i$ for some T_i. A real new class will take away some probability mass from all classes in T and $MT_{n+1} > MT_i \; \forall i = 1 \ldots n$. By a converse argument we should get $MV_{n+1} < MT_i \; \forall i = 1 \ldots n$ for a genuine $(n+1)^{th}$ class because a genuine new class will not have any support for the n-class documents $d \in V$.

True class	MV_i	MT_i
CANA	0.00001526	0.000073
CHINA	0.00001521	0.000093
FRA	0.00001556	0.000077
GFR	0.00001530	0.000111
INDIA	0.00001548	0.000083
NETH	0.00001571	0.000075
RUSS	0.00001607	0.000062
SAFR	0.00001580	0.000071
UK	0.00001621	0.000043
USA	0.00001655	0.000093
AUSTR	**0.00001509**	**0.000121**

Figure 5. Discovering *Australia*

In figure 5 we show some of the 20 values for the above prototyping method. The original label-set had the classes, *USA*, *UK*, *Canada*, and so on. The class *Australia* was hidden in the unlabeled data and had to be discovered. We measured MV_i and MT_i by adding fake classes from *USA*, *UK*, *Canada* etc. The last row shows values of MV_{n+1} and MT_{n+1} when the *Australia* class is really inserted into the label-set using *GenSupp*. We see that MV_{n+1} and MT_{n+1} are respectively minimum and maximum compared to the prototypes.

We note that the differences in values are small, but since the evaluation is on a constant V, we can work out a heuristic which says that the unlabeled data contains a new class when evaluation of candidate new classes satisfies $MV_{n+1} > MV_i$ and $MT_{n+1} < MT_i \; \forall i = 1 \ldots n$.

5. Experiments

Dataset: We conducted Class-Detector experiments with the RCV1 dataset [3]. The RCV1 dataset is a collection of one year of Reuters news stories from August '96 to August '97. The news stories are organized into three unrelated label-sets: *regions*, *topics*, and *industries*. The *topics* and *industries* label-sets are hierarchical while *regions* is a flat label-set. Stories are assigned labels from all three label-sets and multi-labeling within a label-set is common. As we are dealing with the evolving label-set problem, we used the news stories of the first two days. The first day's stories were taken as training data and the second day's stories were taken as unlabeled data for all our Class-Detector experiments. We considered the 20 most populous classes

[3] `http://trec.nist.gov/data/reuters/reuters.html`

each, over all three taxonomies, for these two days. For *topics*, the parent classes CCAT, ECAT, MCAT, GCAT were not chosen; every document in RCV1 is labeled with all labels on the path from root to leaf and since *topics* is a hierarchical label-set we ignored the four top-level classes. This resulted in 4525 documents for regions, 11637 for topics, and 1571 for industries. This selection gave us good varied datasets in terms of variety of classes and sizes for the experiments.

Abstractions: We experimented with the full vocabulary (global *G*), the location (*L*), organization (*O*), person name (*P*) NE tags, and the singular noun (*N*) POS tag, among other NE and POS abstractions. We also tried combined abstraction sets; however due to lack of space, in the rest of this section we report results only with $\{G,P,L\}$ for *regions* and $\{G,O,P\}$ for *topics* and *industries*. We found these to be the most appropriate and understandable abstractions for these datasets.

We used a custom developed named-entity tagger [13] for finding the *P*, *L*, *O* abstractions. We note that this tagging was imperfect and noisy, yet our Class-Detector methods worked well. We used SVMLight [4] for our experiments with SVMs, and a Java implementation of BayesANIL [14]. For HAC we used Peter Kleiweg's clustering software [5] with our own implementation of KL-distance. All experiments were run on a dual-processor Pentium Xeon server running Debian Linux with 2GB RAM.

5.1. Selecting new class documents

For each of the three datasets, we hid one class from the training data (first day stories) and introduced it in the unlabeled data (second day stories). The training data thus had 19 classes and the unlabeled data had 20 classes. We checked if our algorithms could detect this new class from the unlabeled data and suggest a good set of documents comprising this class for user inspection. Our algorithms present a ranked list of suggestions and we measure the precision of these suggestions. Precision is the ratio of correctly suggested new-class documents to the total number of suggestions. We used 20 suggestions for the reported experiments; results with varying number of suggestions were similar. In our opinion, 20 is a good number for the user to be able to judge the existence of a new class fitting into the existing label-set. For each dataset, we report the average precision over all 20 class-detection experiments hiding each class one-by-one. We report results for the following four methods:

- *GenSupp* with 20 suggestions denoted $G20$,
- *SortPrD* with 20 suggestions denoted $P20$,
- *NotaSVM* with 20 suggestions, and
- *DisConf*

In Figure 6, 7, and 8 we show the precision values for the *regions*, *topics* and *industries* datasets with three abstractions each. The precision values in each dataset are averaged over 20 experiments. These graphs reveal interesting results about the various methods and the role of abstractions.

First, when we compare the generative method *GenSupp* G20 with the *SortPrD* baseline P20, we find that G20 is either better or at par with P20 in seven out of nine dataset-abstractions combinations. This illustrates that while the $\Pr(d)$ scores are valuable for detecting new classes, they by itself do not suffice, and it is important to account for coherency of the selected documents in defining a possible new class. The only exception is the *topics* taxonomy where we see that $P20$ outperforms or is marginally better than $G20$. The characteristic of this dataset is that it is hierarchical and inherently multi-labeled. A document with a leaf label, is assigned all labels on the path from the root to the leaf. Our choice of 20 classes in this dataset contained five such parent-child pairs. Such multi-labeling paired the parent and child labels together and the documents of the hidden classes were already present in the original label-set. BayesANIL considered this noisy labeling and automatically assigned low $\Pr(d)$ scores to these documents leading to better performance of P20 over G20. We would like to mention here that *PrDNewClass* did worse than G20 and P20 in most of the cases we tried. *PrDNewClass* seeds a new $(n+1)^{th}$ class with documents sorted on $\Pr(d)$ scores which are not very precise. This new class learned by BayesANIL contains a lot of labeling noise and suggestions based on $\Pr(c_{n+1}, d)$ end up with lower average precision than other generative models.

Second, we find that abstractions do play an important role in some of the taxonomies. For the industries dataset, the *O* and *P* abstractions provide higher precision than *G* which includes all terms. For regions, the person name abstraction *P* provides slightly higher precision than *G* for the *GenSupp* method. We investigated why location name *L* was not the best abstraction for this dataset. We found that the dataset was highly skewed in class distribution. The USA class in the dataset accounted for about half of the documents and these USA documents were also multi-labeled. Hence, since common location names were already seen in the dataset, *L* did not prove to be as good as *P* for this dataset.

Third, in all three cases the discriminative methods (*NotaSVM* and *DisConf*) performed significantly worse than

[4] http://svmlight.joachims.org

[5] http://www.let.rug.nl/~kleiweg/clustering/clustering.html

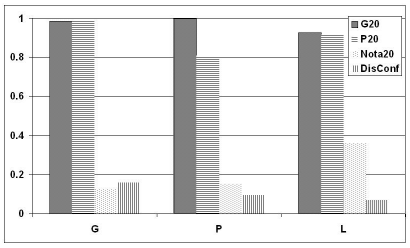

Figure 6. Regions – selecting documents. Y-axis represents average precision

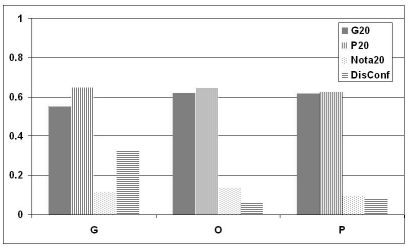

Figure 7. Topics – selecting documents

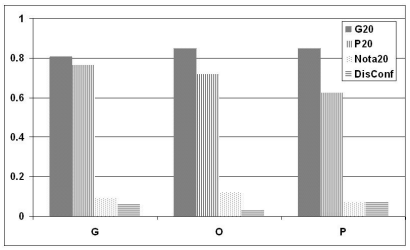

Figure 8. Industries – selecting documents

the generative methods. However, even for the discriminative methods we find that abstractions matter. *NotaSVM* performs slightly better than *DisConf* because it iteratively trains another class over documents predicted NOTA by the original label-set. When we inspected the results of *DisConf*, we found that there were high-precision clusters present in the results of the hierarchical clustering, but we were unable to pick those clusters for suggesting documents. This shows us that one-vs-others SVM output scores, though having similar values for similar documents, do not have large variability and the *DisConf* heuristic for choosing clusters fails. We tried another heuristic in *DisConf* that selected clusters whose average of the least negative (for NOTA as well as non-NOTA) scores was lowest. This heuristic too did not perform much better than the original *DisConf* heuristic. This shows us that it is hard for heuristics based on distances from separator of SVMs to distinguish between new class documents and noisy, multi-labeled mis-classified documents. This also shows that the measure of support is more important for the evolving label-set problem than that of confidence. Discriminative models do not provide such a measure [4] and hence are not very useful for detecting evolving label-sets.

This was a somewhat surprising finding of our experiments because discriminative methods like SVMs are popularly believed to out-perform generative methods for text classification tasks. In the next section, we show that this holds for our dataset too.

5.2. Baseline accuracy

Figure 9 shows the micro-average $F1$ results for all the three datasets (Reg for *regions*, Top for *topics*, and Ind for *industries*) for their chosen three abstractions. SVM and BayesANIL micro-averaged $F1$ values are reported.

SVM outperforms BayesANIL in text classification performance for nearly all dataset and abstraction combinations. It is interesting to see that in the case of the *topics* dataset, TopO and TopP actually do better classification than TopG which looks at the full vocabulary. This affirms

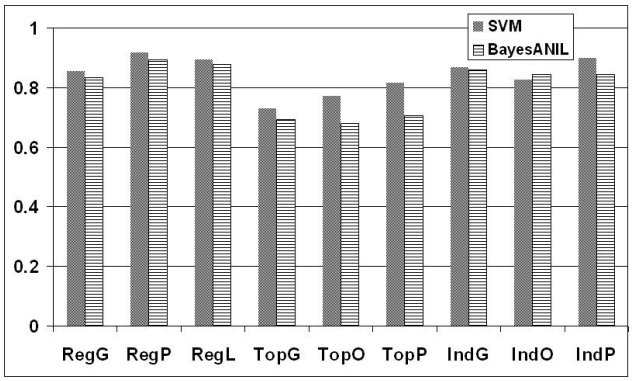

Figure 9. Micro-average F1 for SVM and BayesANIL

our faith in the notion of abstractions – the correct abstractions capture most of the information in the label-set.

5.3. Detecting new classes

We report experiments for detecting new classes according to the MT and MV measures outlined in Section 4. For the *regions* dataset, we experimented with *G* and *L*. For each of the 20 classes, we hid the class in the training data, introduced it in the unlabeled data, and determined the fake-class prototype values of MT and MV. In Figure 10, we report the number of times we successfully triggered a new class detection (out of 20). We see that MT is a better measure than MV. MT_{n+1} is higher than all fake MT_i values more number of times because the fake classes have low support for documents determined to belong to them but which actually come from an existing class. MV performed comparitively poorly. We also see that abstractions perform better than the full vocabulary in trigger new classes and have lesser false negatives.

Regions	MV	MT
G	8	14
L	12	17

Figure 10. Regions – number of times new classes triggered (out of 20)

6. Related Work

Our work is related to the work on Topic Detection and Tracking (TDT) [3, 2, 15] but the problem setting and approaches are different. The aim of TDT is to monitor an online feed of news stories and to detect the first occurrence of a new real world event reported in the news. This is called First Story Detection (FSD) and is followed by tracking further follow up news stories about the event. Most popular techniques at new event detection and tracking (Allan *et al.* [3, 15]) involve a single pass clustering algorithm with well-tuned novelty detection thresholds. Incoming stories are compared to prototypes (average vectors) of existing events. If incoming stories are more than a threshold away from existing events, a new event is spawned. Some of these systems also explore the use of named-entity tags [7, 15] to define more meaningful similarities between documents. This is related to our method of using abstractions, but our notion of abstractions is more general and not limited to a fixed set of NE tags unlike in news stories. In summary, most work on TDT needs to rely on unsupervised clustering techniques using word-based or NE tags-based similarity. In contrast in our classification setting, the set of categories is smaller and known in advance. This makes it possible to project documents in a space that better captures their grouping as far as the set of classes in concerned. Also, most TDT systems cannot handle multi-labeled (multi-event) stories.

Concept drift in classification is another related field of work, but it is quite different from our setting where the set of labels itself changes over time. In concept drift, the distribution of indicative words, and pattern of an *one* class changes over time. A method of dealing with concept drifts in SVMs is provided in Klinkenberg *et al.* [11]. An interesting future work for us would be discovery of new classes in the face of concept drifts of existing classes.

7. Conclusions

We have introduced the evolving label-set problem and presented generative and discriminative methods for dealing with this problem in text classification systems. We introduced the notion of abstractions, which helps the user in understanding label-sets. We use abstractions as a basis for checking the existence of new classes in unlabeled data. We presented the *GenSupp*, *SortPrD*, and *PrDNewClass* algorithms which use state-of-the-art generative models, and the *NotaSVM* and *DisConf* algorithms using discriminative classifiers. An interesting result of our experiments was that though discriminative models were better in text classification performance, generative models outperformed them in Class-Detector experiments.

In future work, we would like to integrate evolving label-set detection in working text classification systems and workbenches like HIClass [8]. In this paper we have considered the introduction of one class at a time. These need to be extended to detect more than one class at time.

References

[1] 3rd workshop on operational text classification. OTC 2003, In conjunction with SIGKDD '03.

[2] J. Allan, V. Lavrenko, and H. Jin. First story detection in TDT is hard. In *Proc. of CIKM 2000*.

[3] J. Allan, R. Papka, and V. Lavrenko. Online new event detection and tracking. In *Proc. of SIGIR '98*.

[4] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. In *Technical Report RC23462, IBM T.J. Watson Research Center*, 2004.

[5] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. In *Proc. of NIPS14, 2002*.

[6] A. El-Hamdouchi and P. Willett. Hierarchic document clustering using ward's method. In *Information Processing and Management*, 1986.

[7] E. Gabrilovich, S. Dumais, and E. Horvitz. Newsjunkie: providing personalized newsfeeds via analysis of information novelty. In *Proc. of WWW '04*, 2004.

[8] S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. Document classification through interactive supervision of document and term labels. In *Proc. of ECML/PKDD '04*.

[9] T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of UAI'99*.

[10] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proc. of ECML-98*.

[11] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proc. of ICML-00*.

[12] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *Proc. of IJCAI-99 Workshop on Machine Learning for Information Filtering*.

[13] G. Ramakrishnan. *Bridging chasms in text mining through Word and Entity Associations*. PhD thesis, IIT Bombay, 2005.

[14] G. Ramakrishnan, K. P. Chitrapura, R. Krishnapuram, and P. Bhattacharya. A model for handling approximate, noisy or incomplete labeling in text classification. In *Proc. of ICML 2005*. `http://www.cse.itib.ac.in/~hare/bayesanil.pdf`.

[15] Y. Yang, J. Zhang, J. Carbonell, and C. Jin. Topic-conditioned novelty detection. In *Proc of SIGKDD '02*.

[16] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *Proc. of SIGIR '03*.