Active Learning with Efficient Feature Weighting Methods for Improving Data Quality and Classification Accuracy

Justin Martineau[1], Lu Chen[2]*, Doreen Cheng[3], and **Amit Sheth[4]**
[1,3] Samsung Research America, Silicon Valley
[1,3] 75 W Plumeria Dr. San Jose, CA 95134 USA
[2,4] Kno.e.sis Center, Wright State University
[2,4] 3640 Colonel Glenn Hwy. Fairborn, OH 45435 USA
[1,3] {justin.m, doreen.c}@samsung.com
[2,4] {chen, amit}@knoesis.org

Abstract

Many machine learning datasets are noisy with a substantial number of mislabeled instances. This noise yields sub-optimal classification performance. In this paper we study a large, low quality annotated dataset, created quickly and cheaply using Amazon Mechanical Turk to crowdsource annotations. We describe computationally cheap feature weighting techniques and a novel non-linear distribution spreading algorithm that can be used to iteratively and interactively correcting mislabeled instances to significantly improve annotation quality at low cost. Eight different emotion extraction experiments on Twitter data demonstrate that our approach is just as effective as more computationally expensive techniques. Our techniques save a considerable amount of time.

1 Introduction

Supervised classification algorithms require annotated data to teach the machine, by example, how to perform a specific task. There are generally two ways to collect annotations of a dataset: through a few expert annotators, or through crowdsourcing services (e.g., Amazon's Mechanical Turk). High-quality annotations can be produced by expert annotators, but the process is usually slow and costly. The latter option is appealing since it creates a large annotated dataset at low cost. In recent years, there have been an increasing number of studies (Su et al., 2007; Kittur et al., 2008; Sheng et al., 2008; Snow et al., 2008; Callison-Burch, 2009) using crowdsourcing for data annotation. However, because annotators that are recruited this way may lack expertise and motivation, the annotations tend to be more noisy and unreliable, which significantly reduces the performance of the classification model. This is a challenge faced by many real world applications – *given a large, quickly and cheaply created, low quality annotated dataset, how can one improve its quality and learn an accurate classifier from it?*

Re-annotating the whole dataset is too expensive. To reduce the annotation effort, it is desirable to have an algorithm that selects the most likely mislabeled examples first for re-labeling. The process of selecting and re-labeling data points can be conducted with multiple rounds to iteratively improve the data quality. This is similar to the strategy of active learning. The basic idea of active learning is to learn an accurate classifier using less training data. An active learner uses a small set of labeled data to iteratively select the most informative instances from a large pool of unlabeled data for human annotators to label (Settles, 2010). In this work, we borrow the idea of active learning to interactively and iteratively correct labeling errors.

The crucial step is to *effectively and efficiently* select the most likely mislabeled instances. An intuitive idea is to design algorithms that classify the data points and rank them according to the decreasing confidence scores of their labels. The data points with the highest confidence scores but conflicting preliminary labels are most likely mislabeled. The algorithm should be computationally cheap as well as accurate, so it fits well with active learning and other problems that require frequent iterations on large datasets. Specifically, we propose a novel non-linear distribution spreading algorithm, which first uses Delta IDF technique (Martineau and Finin, 2009) to weight features, and then leverages the distribution of Delta IDF scores of a feature across different classes to efficiently recognize discriminative features for the classification task in the presence of mislabeled data. The idea is that some effective fea-

*This author's research was done during an internship with Samsung Research America.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 1104–1112,
Baltimore, Maryland, USA, June 23-25 2014.

tures may be subdued due to label noise, and the proposed techniques are capable of counteracting such effect, so that the performance of classification algorithms could be less affected by the noise. With the proposed algorithm, the active learner becomes more accurate and resistant to label noise, thus the mislabeled data points can be more easily and accurately identified.

We consider emotion analysis as an interesting and challenging problem domain of this study, and conduct comprehensive experiments on Twitter data. We employ Amazon's Mechanical Turk (AMT) to label the emotions of Twitter data, and apply the proposed methods to the AMT dataset with the goals of improving the annotation quality at low cost, as well as learning accurate emotion classifiers. Extensive experiments show that, the proposed techniques are as effective as more computational expensive techniques (e.g, Support Vector Machines) but require significantly less time for training/running, which makes it well-suited for active learning.

2 Related Work

Research on handling noisy dataset of mislabeled instances has focused on three major groups of techniques: (1) noise tolerance, (2) noise elimination, and (3) noise correction.

Noise tolerance techniques aim to improve the learning algorithm itself to avoid over-fitting caused by mislabeled instances in the training phase, so that the constructed classifier becomes more noise-tolerant. Decision tree (Mingers, 1989; Vannoorenberghe and Denoeux, 2002) and boosting (Jiang, 2001; Kalaia and Servediob, 2005; Karmaker and Kwek, 2006) are two learning algorithms that have been investigated in many studies. Mingers (1989) explores pruning methods for identifying and removing unreliable branches from a decision tree to reduce the influence of noise. Vannoorenberghe and Denoeux (2002) propose a method based on belief decision trees to handle uncertain labels in the training set. Jiang (2001) studies some theoretical aspects of regression and classification boosting algorithms in dealing with noisy data. Kalaia and Servediob (2005) present a boosting algorithm which can achieve arbitrarily high accuracy in the presence of data noise. Karmaker and Kwek (2006) propose a modified AdaBoost algorithm – ORBoost, which minimizes the impact of outliers and becomes more tolerant to class label noise. One of the main disadvantages of noise tolerance techniques is that they are learning algorithm-dependent. In contrast, noise elimination/correction approaches are more generic and can be more easily applied to various problems.

A large number of studies have explored noise elimination techniques (Brodley and Friedl, 1999; Verbaeten and Van Assche, 2003; Zhu et al., 2003; Muhlenbach et al., 2004; Guan et al., 2011), which identifies and removes mislabeled examples from the dataset as a pre-processing step before building classifiers. One widely used approach (Brodley and Friedl, 1999; Verbaeten and Van Assche, 2003) is to create an ensemble classifier that combines the outputs of multiple classifiers by either majority vote or consensus, and an instance is tagged as mislabeled and removed from the training set if it is classified into a different class than its training label by the ensemble classifier. The similar approach is adopted by Guan et al. (2011) and they further demonstrate that its performance can be significantly improved by utilizing unlabeled data. To deal with the noise in large or distributed datasets, Zhu et al. (2003) propose a partition-based approach, which constructs classification rules from each subset of the dataset, and then evaluates each instance using these rules. Two noise identification schemes, majority and non-objection, are used to combine the decision from each set of rules to decide whether an instance is mislabeled. Muhlenbach et al. (2004) propose a different approach, which represents the proximity between instances in a geometrical neighborhood graph, and an instance is considered suspect if in its neighborhood the proportion of examples of the same class is not significantly greater than in the dataset itself.

Removing mislabeled instances has been demonstrated to be effective in increasing the classification accuracy in prior studies, but there are also some major drawbacks. For example, useful information can be removed with noise elimination, since annotation errors are likely to occur on ambiguous instances that are potentially valuable for learning algorithms. In addition, when the noise ratio is high, there may not be adequate amount of data remaining for building an accurate classifier. The proposed approach does not suffer these limitations.

Instead of eliminating the mislabeled examples

from training data, some researchers (Zeng and Martinez, 2001; Rebbapragada et al., 2012; Laxman et al., 2013) propose to correct labeling errors either with or without consulting human experts. Zeng and Martinez (2001) present an approach based on backpropagation neural networks to automatically correct the mislabeled data. Laxman et al. (2012) propose an algorithm which first trains individual SVM classifiers on several small, class-balanced, random subsets of the dataset, and then reclassifies each training instance using a majority vote of these individual classifiers. However, the automatic correction may introduce new noise to the dataset by mistakenly changing a correct label to a wrong one.

In many scenarios, it is worth the effort and cost to fix the labeling errors by human experts, in order to obtain a high quality dataset that can be reused by the community. Rebbapragada et al. (2012) propose a solution called Active Label Correction (ALC) which iteratively presents the experts with small sets of suspected mislabeled instances at each round. Our work employs a similar framework that uses active learning for data cleaning.

In Active Learning (Settles, 2010) a small set of labeled data is used to find documents that should be annotated from a large pool of unlabeled documents. Many different strategies have been used to select the best points to annotate. These strategies can be generally divided into two groups: (1) selecting points in poorly sampled regions, and (2) selecting points that will have the greatest impact on models that were constructed using the dataset.

Active learning for data cleaning differs from traditional active learning because the data already has low quality labels. It uses the difference between the low quality label for each data point and a prediction of the label using supervised machine learning models built upon the low quality labels. Unlike the work in (Rebbapragada et al., 2012), this paper focuses on developing algorithms that can enhance the ability of active learner on identifying labeling errors, which we consider as a key challenge of this approach but ALC has not addressed.

3 An Active Learning Framework for Label Correction

Let $\hat{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$ be a dataset of binary labeled instances, where the instance x_i belongs to domain X, and its label $y_i \in \{-1, +1\}$. $\hat{D}$ contains an unknown number of mislabeled data points. The problem is to obtain a high-quality dataset D by fixing labeling errors in $\hat{D}$, and learn an accurate classifier C from it.

Algorithm 1 illustrates an active learning approach to the problem. This algorithm takes the noisy dataset $\hat{D}$ as input. The training set T is initialized with the data in $\hat{D}$ and then updated each round with new labels generated during re-annotation. Data sets S_r and S are used to maintain the instances that have been selected for re-annotation in the whole process and in the current iteration, respectively.

Data: noisy data $\hat{D}$
Result: cleaned data D, classifier C
Initialize training set $T = \hat{D}$;
Initialize re-annotated data sets $S_r = \emptyset$;
$S = \emptyset$;
repeat
  Train classifier C using T ;
  Use C to select a set S of m suspected mislabeled instances from T ;
  Experts re-annotate the instances in $S - (S_r \cap S)$;
  Update T with the new labels in S ;
  $S_r = S_r \cup S$; $S = \emptyset$;
until *for I iterations*;
$D = T$;

Algorithm 1: Active Learning Approach for Label Correction

In each iteration, the algorithm trains classifiers using the training data in T. In practice, we apply k-fold cross-validation. We partition T into k subsets, and each time we keep a different subset as testing data and train a classifier using the other $k-1$ subsets of data. This process is repeated k times so that we get a classifier for each of the k subsets. The goal is to use the classifiers to efficiently and accurately seek out the most likely mislabeled instances from T for expert annotators to examine and re-annotate. When applying a classifier to classify the instances in the corresponding data subset, we get the probability about how likely one instance belongs to a class. The top m instances with the highest probabilities belonging to some class but conflicting preliminary labels are selected as the most likely errors for annotators to fix. During the re-annotation process we keep the old labels hidden to prevent that information from

biasing annotators' decisions. Similarly, we keep the probability scores hidden while annotating.

This process is done with multiple iterations of training, sampling, and re-annotating. We maintain the re-annotated instances in S_r to avoid annotating the same instance multiple times. After each round of annotation, we compare the old labels to the new labels to measure the degree of impact this process is having on the dataset. We stop re-annotating on the Ith round after we decide that the reward for an additional round of annotation is too low to justify.

4 Feature Weighting Methods

Building the classifier C that allows the most likely mislabeled instances to be selected and annotated is the essence of the active learning approach. There are two main goals of developing this classifier: (1) accurately predicting the labels of data points and ranking them based on prediction confidence, so that the most likely errors can be effectively identified; (2) requiring less time on training, so that the saved time can be spent on correcting more labeling errors. Thus we aim to build a classifier that is both accurate and time efficient.

Labeling noise affects the classification accuracy. One possible reason is that some effective features that should be given high weights are inhibited in the training phase due to the labeling errors. For example, emoticon ":D" is a good indicator for emotion *happy*, however, if by mistake many instances containing this emoticon are not correctly labeled as *happy*, this class-specific feature would be underestimated during training. Following this idea, we develop computationally cheap feature weighting techniques to counteract such effect by boosting the weight of discriminative features, so that they would not be subdued and the instances with such features would have higher chance to be correctly classified.

Specifically, we propose a non-linear distribution spreading algorithm for feature weighting. This algorithm first utilizes Delta IDF to weigh the features, and then non-linearly spreads out the distribution of features' Delta IDF scores to exaggerate the weight of discriminative features. We first introduce Delta-IDF technique, and then describe our algorithm of distribution spreading. Since we focus on n-gram features, we use the words *feature* and *term* interchangeably in this paper.

4.1 Delta IDF Weighting Scheme

Different from the commonly used TF (term frequency) or TF.IDF (term frequency.inverse document frequency) weighting schemes, Delta IDF treats the positive and negative training instances as two separate corpora, and weighs the terms by how biased they are to one corpus. The more biased a term is to one class, the higher (absolute value of) weight it will get. Delta IDF boosts the importance of terms that tend to be class-specific in the dataset, since they are usually effective features in distinguishing one class from another.

Each training instance (e.g., a document) is represented as a feature vector: $x_i = (w_{1,i}, ..., w_{|V|,i})$, where each dimension in the vector corresponds to a n-gram term in vocabulary $V = \{t_1, ..., t_{|V|}\}$, $|V|$ is the number of unique terms, and $w_{j,i} (1 \leq j \leq |V|)$ is the weight of term t_j in instance x_i. Delta IDF (Martineau and Finin, 2009) assigns score Δ_idf_j to term t_j in V as:

$$\Delta\_idf_j = \log \frac{(N+1)(P_j+1)}{(N_j+1)(P+1)} \quad (1)$$

where P (or N) is the number of positively (or negatively) labeled training instances, P_j (or N_j) is the number of positively (or negatively) labeled training instances with term t_j. Simple add-one smoothing is used to smooth low frequency terms and prevent dividing by zero when a term appears in only one corpus. We calculate the Delta IDF score of every term in V, and get the Delta IDF weight vector $\Delta = (\Delta\_idf_1, ..., \Delta\_idf_{|V|})$ for all terms.

When the dataset is imblanced, to avoid building a biased model, we down sample the majority class before calculating the Delta IDF score and then use the a bias balancing procedure to balance the Delta IDF weight vector. This procedure first divides the Delta IDF weight vector to two vectors, one of which contains all the features with positive scores, and the other of which contains all the features with negative scores. It then applies L2 normalization to each of the two vectors, and add them together to create the final vector.

For each instance, we can calculate the TF.Delta-IDF score as its weight:

$$w_{j,i} = tf_{j,i} \times \Delta\_idf_j \quad (2)$$

where $tf_{j,i}$ is the number of times term t_j occurs in document x_i, and Δ_idf_j is the Delta IDF score of t_j.

4.2 A Non-linear Distribution Spreading Algorithm

Delta IDF technique boosts the weight of features with strong discriminative power. The model's ability to discriminate at the feature level can be further enhanced by leveraging the distribution of feature weights across multiple classes, e.g., multiple emotion categories *funny*, *happy*, *sad*, *exciting*, *boring*, etc.. The distinction of multiple classes can be used to further force feature bias scores apart to improve the identification of class-specific features in the presence of labeling errors.

Let L be a set of target classes, and $|L|$ be the number of classes in L. For each class $l \in L$, we create a binary labeled dataset $\hat{D}^l$. Let V^l be the vocabulary of dataset $\hat{D}^l$, V be the vocabulary of all datasets, and $|V|$ is the number of unique terms in V. Using Formula (1) and dataset $\hat{D}^l$, we get the Delta IDF weight vector for each class l: $\Delta^l = (\Delta\_idf_1^l, ..., \Delta\_idf_{|V|}^l)$. Note that $\Delta\_idf_j^l = 0$ for any term $t_j \in V - V^l$. For a class u, we calculate the spreading score $spread_j^u$ of each feature $t_j \in V$ using a non-linear distribution spreading formula as following (where s is the configurable spread parameter):

$$spread_j^u = \Delta\_idf_j^u \times \frac{\sum_{l \in L-u} |\Delta\_idf_j^u - \Delta\_idf_j^l|^s}{|L| - 1} \quad (3)$$

For any term $t_j \in V$, we can get its Delta IDF score on a class l. The distribution of Delta IDF scores of t_j on all classes in L is represented as $\delta_j = \{\Delta\_idf_j^1, ..., \Delta\_idf_j^{|L|}\}$.

The mechanism of Formula (3) is to non-linearly spread out the distribution, so that the importance of class-specific features can be further boosted to counteract the effect of noisy labels. Specifically, according to Formula (3), a high (absolute value of) spread score indicates that the Delta IDF score of that term on that class is high and deviates greatly from the scores on other classes. In other words, our algorithm assigns high spread score (absolute value) to a term on a class for which the term has strong discriminative power and very specific to that class compared with to other classes. When the dataset is imbalanced, we apply the similar bias balancing procedure as described in Section 4.1 to the spreading model.

While these feature weighting models can be used to score and rank instances for data cleaning, better classification and regression models can be built by using the feature weights generated by these models as a pre-weight on the data points for other machine learning algorithms.

5 Experiments

We conduct experiments on a Twitter dataset that contains tweets about TV shows and movies. The goal is to extract consumers' emotional reactions to multimedia content, which has broad commercial applications including targeted advertising, intelligent search, and recommendation. To create the dataset, we collected 2 billion unique tweets using Twitter API queries for a list of known TV shows and movies on IMDB. Spam tweets were filtered out using a set of heuristics and manually crafted rules. From the set of 2 billion tweets we randomly selected a small subset of 100K tweets about the 60 most highly mentioned TV shows and movies in the dataset. Tweets were randomly sampled for each show using the round robin algorithm. Duplicates were not allowed. This samples an equal number of tweets for each show. We then sent these tweets to Amazon Mechanical Turk for annotation.

We defined our own set of emotions to annotate. The widely accepted emotion taxonomies, including Ekmans Basic Emotions (Ekman, 1999), Russells Circumplex model (Russell and Barrett, 1999), and Plutchiks emotion wheel (Plutchik, 2001), did not fit well for TV shows and Movies. For example, the emotion expressed by laughter is a very important emotion for TV shows and movies, but this emotion is not covered by the taxonomies listed above. After browsing through the raw dataset, reviewing the literature on emotion analysis, and considering the TV and movie problem domain, we decided to focus on eight emotions: *funny*, *happy*, *sad*, *exciting*, *boring*, *angry*, *fear*, and *heartwarming*.

Emotion annotation is a non-trivial task that is typically time-consuming, expensive and error-prone. This task is difficult because: (1) There are multiple emotions to annotate. In this work, we annotate eight different emotions. (2) Emotion expressions could be subtle and ambiguous and thus are easy to miss when labeling quickly. (3) The dataset is very imbalanced, which increases the problem of confirmation bias. As minority classes, emotional tweets can be easily missed because the last X tweets are all not emotional, and the annota-

	Funny	Happy	Sad	Exciting	Boring	Angry	Fear	Heartwarming
# Pos.	1,324	405	618	313	209	92	164	24
# Neg.	88,782	95,639	84,212	79,902	82,443	57,326	46,746	15,857
# Total	90,106	96,044	84,830	80,215	82,652	57,418	46,910	15,881

Table 1: Amazon Mechanical Turk annotation label counts.

	Funny	Happy	Sad	Exciting	Boring	Angry	Fear	Heartwarming
# Pos.	1,781	4,847	788	1,613	216	763	285	326
# Neg.	88,277	91,075	84,031	78,573	82,416	56,584	46,622	15,542
# Total[1]	90,058	95,922	84,819	80,186	82,632	57,347	46,907	15,868

Table 2: Ground truth annotation label counts for each emotion.[2]

tors do not expect the next one to be either. Due to these reasons, there is a lack of sufficient and high quality labeled data for emotion research. Some researchers have studied harnessing Twitter hashtags to automatically create an emotion annotated dataset (Wang et al., 2012).

In order to evaluate our approach in real world scenarios, instead of creating a high quality annotated dataset and then introducing artificial noise, we followed the common practice of crowdsoucing, and collected emotion annotations through Amazon Mechanical Turk (AMT). This AMT annotated dataset was used as the low quality dataset $\hat{D}$ in our evaluation. After that, the same dataset was annotated independently by a group of expert annotators to create the ground truth. We evaluate the proposed approach on two factors, the effectiveness of the models for emotion classification, and the improvement of annotation quality provided by the active learning procedure. We first describe the AMT annotation and ground truth annotation, and then discuss the baselines and experimental results.

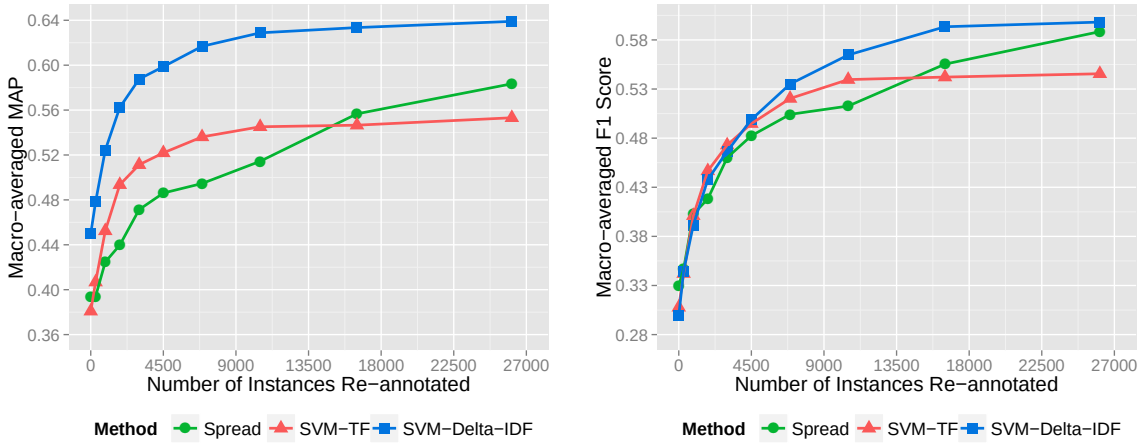
Amazon Mechanical Turk Annotation: we posted the set of 100K tweets to the workers on AMT for emotion annotation. We defined a set of annotation guidelines, which specified rules and examples to help annotators determine when to tag a tweet with an emotion. We applied substantial quality control to our AMT workers to improve the initial quality of annotation following the common practice of crowdsourcing. Each tweet was annotated by at least two workers. We used a series of tests to identify bad workers. These tests include (1) identifying workers with poor pairwise agreement, (2) identifying workers with poor performance on English language annotation, (3) identifying workers that were annotating at unrealistic speeds, (4) identifying workers with near random annotation distributions, and (5) identifying workers that annotate each tweet for a given TV show the same (or nearly the same) way. We manually inspected any worker with low performance on any of these tests before we made a final decision about using any of their annotations.

For further quality control, we also gathered additional annotations from additional workers for tweets where only one out of two workers identified an emotion. After these quality control steps we defined minimum emotion annotation thresholds to determine and assign preliminary emotion labels to tweets. Note that some tweets were discarded as mixed examples for each emotion based upon thresholds for how many times they were tagged, and it resulted in different number of tweets in each emotion dataset. See Table 1 for the statistics of the annotations collected from AMT.

Ground Truth Annotation: After we obtained the annotated dataset from AMT, we posted the same dataset (without the labels) to a group of expert annotators. The experts followed the same annotation guidelines, and each tweet was labeled by at least two experts. When there was a disagreement between two experts, they discussed to reach an agreement or gathered additional opinion from another expert to decide the label of a tweet. We used this annotated dataset as ground truth. See Table 2 for the statistics of the ground truth annotations. Compared with the ground truth, many emotion bearing tweets were missed by the AMT annotators, despite the quality control we applied. It demonstrates the challenge of annotation by crowdsourcing. The imbalanced class distribution

[1]The total number of tweets is lower than the AMT dataset because the experts removed some off-topic tweets.

[2]Expert annotators had a Kappa agreement score of 0.639 before meeting to resolve their differences.



(a) Macro-Averaged MAP

(b) Macro-Averaged F1 Score

Figure 1: Performance comparison of mislabeled instance selection methods. Classifiers become more accurate as more instances are re-annotated. Spread achieves comparable performance with SVMs in terms of both MAP and F1 Score.

aggravates the confirmation bias – the minority class examples are especially easy to miss when labeling quickly due to their rare presence in the dataset.

Evaluation Metric: We evaluated the results with both Mean Average Precision (MAP) and F1 Score. Average Precision (AP) is the average of the algorithm's precision at every position in the confidence ranked list of results where a true emotional document has been identified. Thus, AP places extra emphasis on getting the front of the list correct. MAP is the mean of the average precision scores for each ranked list. This is highly desirable for many practical application such as intelligent search, recommendation, and target advertising where users almost never see results that are not at the top of the list. F1 is a widely-used measure of classification accuracy.

Methods: We evaluated the overall performance relative to the common SVM bag of words approach that can be ubiquitously found in text mining literature. We implemented the following four classification methods:

- **Delta-IDF**: Takes the dot product of the Delta IDF weight vector (Formula 1) with the document's term frequency vector.

- **Spread**: Takes the dot product of the distribution spread weight vector (Formula 3) with the document's term frequency vector. For all the experiments, we used spread parameter $s = 2$.

- **SVM-TF**: Uses a bag of words SVM with term frequency weights.

- **SVM-Delta-IDF**: Uses a bag of words SVM classification with TF.Delta-IDF weights (Formula 2) in the feature vectors before training or testing an SVM.

We employed each method to build the active learner C described in Algorithm 1. We used standard bag of *unigram* and *bigram* words representation and *topic-based fold cross validation*. Since in real world applications people are primarily concerned with how well the algorithm will work for new TV shows or movies that may not be included in the training data, we defined a test fold for each TV show or movie in our labeled data set. Each test fold corresponded to a training fold containing all the labeled data from all the other TV shows and movies. We call it topic-based fold cross validation.

We built the SVM classifiers using LIBLINEAR (Fan et al., 2008) and applied its L2-regularized support vector regression model. Based on the dot product or SVM regression scores, we ranked the tweets by how strongly they express the emotion. We selected the top m tweets with the highest dot product or regression scores but conflicting preliminary AMT labels as the suspected mislabeled instances for re-annotation, just as described in Algorithm 1. For the experimental purpose, the re-annotation was done by assigning the ground truth labels to the selected instances. Since the dataset is highly imbalanced, we applied the *under-sampling strategy* when training the classifiers.

Figure 1 compares the performance of different approaches in each iteration after a certain number of potentially mislabeled instances are re-

annotated. The X axis shows the total number of data points that have been examined for each emotion so far till the current iteration (i.e., 300, 900, 1800, 3000, 4500, 6900, 10500, 16500, and 26100). We reported both the macro-averaged MAP (Figure 1a) and the macro-averaged F1 Score (Figure 1b) on eight emotions as the overall performance of three competitive methods – Spread, SVM-Delta-IDF and SVM-TF. We have also conducted experiments using Delta-IDF, but its performance is low and not comparable with the other three methods.

Generally, Figure 1 shows consistent performance gains as more labels are corrected during active learning. In comparison, SVM-Delta-IDF significantly outperforms SVM-TF with respect to both MAP and F1 Score. SVM-TF achieves higher MAP and F1 Score than Spread at the first few iterations, but then it is beat by Spread after 16,500 tweets had been selected and re-annotated till the eighth iteration. Overall, at the end of the active learning process, Spread outperforms SVM-TF by 3.03% the MAP score (and by 4.29% the F1 score), and SVM-Delta-IDF outperforms SVM-TF by 8.59% the MAP score (and by 5.26% the F1 score). Spread achieves a F1 Score of 58.84%, which is quite competitive compared to 59.82% achieved by SVM-Delta-IDF, though SVM-Delta-IDF outperforms Spread with respect to MAP.

Spread and Delta-IDF are superior with respect to the time efficiency. Figure 2 shows the average training time of the four methods on eight emotions. The time spent training SVM-TF classifiers is twice that of SVM-Delta-IDF classifiers, 12 times that of Spread classifiers, and 31 times that of Delta-IDF classifiers. In our experiments, on average, it took 258.8 seconds to train a SVM-TF classifier for one emotion. In comparison, the average training time of a Spread classifier was only 21.4 seconds, and it required almost no parameter tuning. In total, our method Spread saved up to $(258.8 - 21.4) * 9 * 8 = 17092.8$ seconds (4.75 hours) over nine iterations of active learning for all the eight emotions. This is enough time to re-annotate thousands of data points.

The other important quantity to measure is annotation quality. One measure of improvement for annotation quality is the number of mislabeled instances that can be fixed after a certain number of active learning iterations. Better methods can fix more labels with fewer iterations.

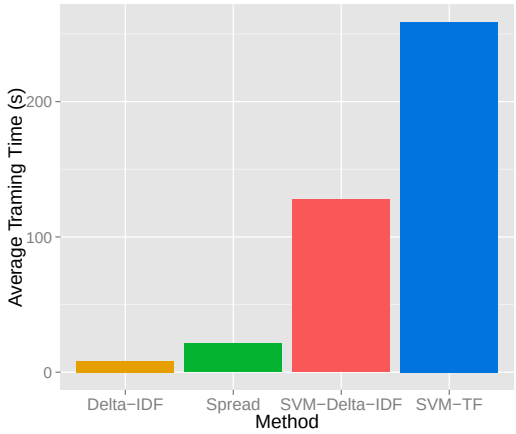


Figure 2: Average training time on eight emotions. Spread requires only one-twelfth of the time spent to training an SVM-TF classifier. Note that the time spent tuning the SVM's parameters has not been included, but is considerable. Compared with such computationally expensive methods, Spread is more appropriate for use with active learning.

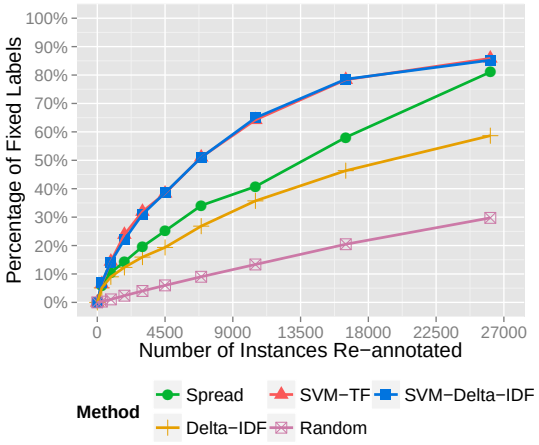


Figure 3: Accumulated average percentage of fixed labels on eight emotions. Spreading the feature weights reduces the number of data points that must be examined in order to correct the mislabeled instances. SVMs require slightly fewer points but take far longer to build.

Besides the four methods, we also implemented a random baseline (Random) which randomly selected the specified number of instances for re-annotation in each round. We compared the improved dataset with the final ground truth at the end of each round to monitor the progress. Figure 3 reports the accumulated average percentage of corrected labels on all emotions in each iteration of the active learning process.

According to the figure, SVM-Delta-IDF and SVM-TF are the most advantageous methods, followed by Spread and Delta-IDF. After the last iteration, SVM-Delta-IDF, SVM-TF, Spread and Delta-IDF has fixed 85.23%, 85.85%, 81.05% and 58.66% of the labels, respectively, all of which significantly outperform the Random baseline (29.74%).

6 Conclusion

In this paper, we explored an active learning approach to improve data annotation quality for classification tasks. Instead of training the active learner using computationally expensive techniques (e.g., SVM-TF), we used a novel non-linear distribution spreading algorithm. This algorithm first weighs the features using the Delta-IDF technique, and then non-linearly spreads out the distribution of the feature scores to enhance the model's ability to discriminate at the feature level. The evaluation shows that our algorithm has the following advantages: (1) It intelligently ordered the data points for annotators to annotate the most likely errors first. The accuracy was at least comparable with computationally expensive baselines (e.g. SVM-TF). (2) The algorithm trained and ran much faster than SVM-TF, allowing annotators to finish more annotations than competitors. (3) The annotation process improved the dataset quality by positively impacting the accuracy of classifiers that were built upon it.

References

Carla E Brodley and Mark A Friedl. 1999. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167.

Chris Callison-Burch. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of EMNLP*, pages 286–295. ACL.

Paul Ekman. 1999. Basic emotions. *Handbook of cognition and emotion*, 4:5–60.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Donghai Guan, Weiwei Yuan, Young-Koo Lee, and Sungyoung Lee. 2011. Identifying mislabeled training data with the aid of unlabeled data. *Applied Intelligence*, 35(3):345–358.

Wenxin Jiang. 2001. Some theoretical aspects of boosting in the presence of noisy data. In *Proceedings of ICML*. Citeseer.

Adam Tauman Kalaia and Rocco A Servediob. 2005. Boosting in the presence of noise. *Journal of Computer and System Sciences*, 71:266–290.

Amitava Karmaker and Stephen Kwek. 2006. A boosting approach to remove class label noise. *International Journal of Hybrid Intelligent Systems*, 3(3):169–177.

Aniket Kittur, Ed H Chi, and Bongwon Suh. 2008. Crowdsourcing user studies with mechanical turk. In *Proceedings of CHI*, pages 453–456. ACM.

Srivatsan Laxman, Sushil Mittal, and Ramarathnam Venkatesan. 2013. Error correction in learning using svms. *arXiv preprint arXiv:1301.2012*.

Justin Martineau and Tim Finin. 2009. Delta tfidf: An improved feature space for sentiment analysis. In *Proceedings of ICWSM*.

John Mingers. 1989. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243.

Fabrice Muhlenbach, Stéphane Lallich, and Djamel A Zighed. 2004. Identifying and handling mislabelled instances. *Journal of Intelligent Information Systems*, 22(1):89–109.

Robert Plutchik. 2001. The nature of emotions. *American Scientist*, 89(4):344–350.

Umaa Rebbapragada, Carla E Brodley, Damien Sulla-Menashe, and Mark A Friedl. 2012. Active label correction. In *Proceedings of ICDM*, pages 1080–1085. IEEE.

James A Russell and Lisa Feldman Barrett. 1999. Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. *Journal of personality and social psychology*, 76(5):805.

Burr Settles. 2010. Active learning literature survey. *Technical Report 1648, University of Wisconsin, Madison*.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. 2008. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of KDD*, pages 614–622. ACM.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. 2008. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of EMNLP*, pages 254–263.

Qi Su, Dmitry Pavlov, Jyh-Herng Chow, and Wendell C Baker. 2007. Internet-scale collection of human-reviewed data. In *Proceedings of WWW*, pages 231–240. ACM.

P Vannoorenberghe and T Denoeux. 2002. Handling uncertain labels in multiclass problems using belief decision trees. In *Proceedings of IPMU*, volume 3, pages 1919–1926.

Sofie Verbaeten and Anneleen Van Assche. 2003. Ensemble methods for noise elimination in classification problems. In *Multiple classifier systems*, pages 317–325. Springer.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing twitter" big data" for automatic emotion identification. In *Proceedings of SocialCom*, pages 587–592. IEEE.

Xinchuan Zeng and Tony R Martinez. 2001. An algorithm for correcting mislabeled data. *Intelligent data analysis*, 5(6):491–502.

Xingquan Zhu, Xindong Wu, and Qijun Chen. 2003. Eliminating class noise in large datasets. In *Proceedings of ICML*, pages 920–927.