Text Classification without Negative Examples Revisit

Gabriel Pui Cheong Fung, Jeffrey X. Yu, *Member*, *IEEE Computer Society*,
Hongjun Lu, and Philip S. Yu, *Fellow*, *IEEE*

Abstract—Traditionally, building a classifier requires two sets of examples: positive examples and negative examples. This paper studies the problem of building a text classifier using positive examples (P) and unlabeled examples (U). The unlabeled examples are mixed with both positive and negative examples. Since no negative example is given explicitly, the task of building a reliable text classifier becomes far more challenging. Simply treating all of the unlabeled examples as negative examples and building a classifier thereafter is undoubtedly a poor approach to tackling this problem. Generally speaking, most of the studies solved this problem by a two-step heuristic: First, extract negative examples (N) from U. Second, build a classifier based on P and N. Surprisingly, most studies did not try to extract positive examples from U. Intuitively, enlarging P by P' (positive examples extracted from U) and building a classifier thereafter should enhance the effectiveness of the classifier. Throughout our study, we find that extracting P' is very difficult. A document in U that possesses the features exhibited in P does not necessarily mean that it is a positive example, and vice versa. The very large size of and very high diversity in U also contribute to the difficulties of extracting P'. In this paper, we propose a labeling heuristic called *PNLH* to tackle this problem. *PNLH* aims at extracting high quality positive examples and negative examples from U and can be used on top of any existing classifiers. Extensive experiments based on several benchmarks are conducted. The results indicated that *PNLH* is highly feasible, especially in the situation where $|P|$ is extremely small.

Index Terms—Data mining, text categorization, partially supervised learning, labeling unlabeled data.

1 Introduction

Text *classification* is a *supervised learning task* where its task is to assign a Boolean value to each pair $(d_j, k_i) \in \mathcal{D} \times \mathcal{K}$, where $\mathcal{D}$ is a domain of documents and $\mathcal{K}$ is a set of predefined categories. The task is to approximate the true function $\phi : \mathcal{D} \times \mathcal{K} \rightarrow \{1, 0\}$ by means of a function $\widehat{\phi} : \mathcal{D} \times \mathcal{K} \rightarrow \{1, 0\}$ such that ϕ and $\widehat{\phi}$ *coincide as much as possible* [20]. The function $\widehat{\phi}$ is called a classifier. The coincidence is measured by *effectiveness*, which is also known as *the quality of the classifier*.

A classifier can be built by training it systematically using a set of training documents $\mathcal{D}$, where *all* of the documents belonging to $\mathcal{D}$ are labeled according to $\mathcal{K}$. Specifically, given a category $k \in \mathcal{K}$, in order to build a classifier for k, we have to determine which of the documents in $\mathcal{D}$ belong to k and which of the documents in $\mathcal{D}$ do not belong to k. The documents that belong to k are labeled as *positive training instances* ($\mathcal{P}$) and the documents that do not belong to k are labeled as *negative training instances* ($\mathcal{N}$). Fig. 1a shows a typical framework for building a classifier.

Note that the effectiveness of the classifier is strongly affected by the precision of labeling. If the documents in ($\mathcal{D}$) are mislabeled, then the classifier built thereafter will be strongly biased, and poor quality must be resulted. Unfortunately, labeling documents is a time consuming and labor intensive task, and it is the bottleneck of constructing a reliable classifier.

Due to the rapid growth of information available, the task of *accurately* labeling *all* of the training documents as positive examples and negative examples becomes very difficult, if not infeasible. In order to overcome this bottleneck, some researchers attempt to divide the training documents $\mathcal{D}$ into three sets (a *small* set of positive examples, a *small* set of negative examples, and a *large* set of unlabeled examples), and then build a classifier using these three sets of documents [1], [2], [6], [16], [26].

However, obtaining a set of negative examples is sometimes still very expensive. Recently, a new direction of building classifiers is recognized where the classifiers are built using only a small set of positive examples (P) and a large set of unlabeled examples (U) [5], [9], [13], [11], [10], [12], [23], [25], [24]. Note that *no* negative examples are given. This kind of problem is sometimes known as *partially supervised learning* [10]. The characteristics of this partially supervised learning are summarized as follows: 1) The size of the given positive examples (P) is so small that it may not be possible to represent the feature distribution of *all* positive examples, 2) the unlabeled examples (U) are mixed with both positive and negative examples, and 3) no negative example is given.

Generally speaking, the existing approaches that target solving this problem use a two-step heuristics as shown in Fig. 1b. In the first step, some negative examples (N) are extracted from the unlabeled examples (U). In the second

- *G.P.C. Fung and J.X. Yu are with the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, China. E-mail: {pcfung, yu}@se.cuhk.edu.hk.*
- *P.S. Yu is with the IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, New York. E-mail: psyu@us.ibm.com.*

Manuscript received 7 Feb. 2005; revised 9 June 2005; accepted 27 June 2005; published online 18 Nov. 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0052-0205.

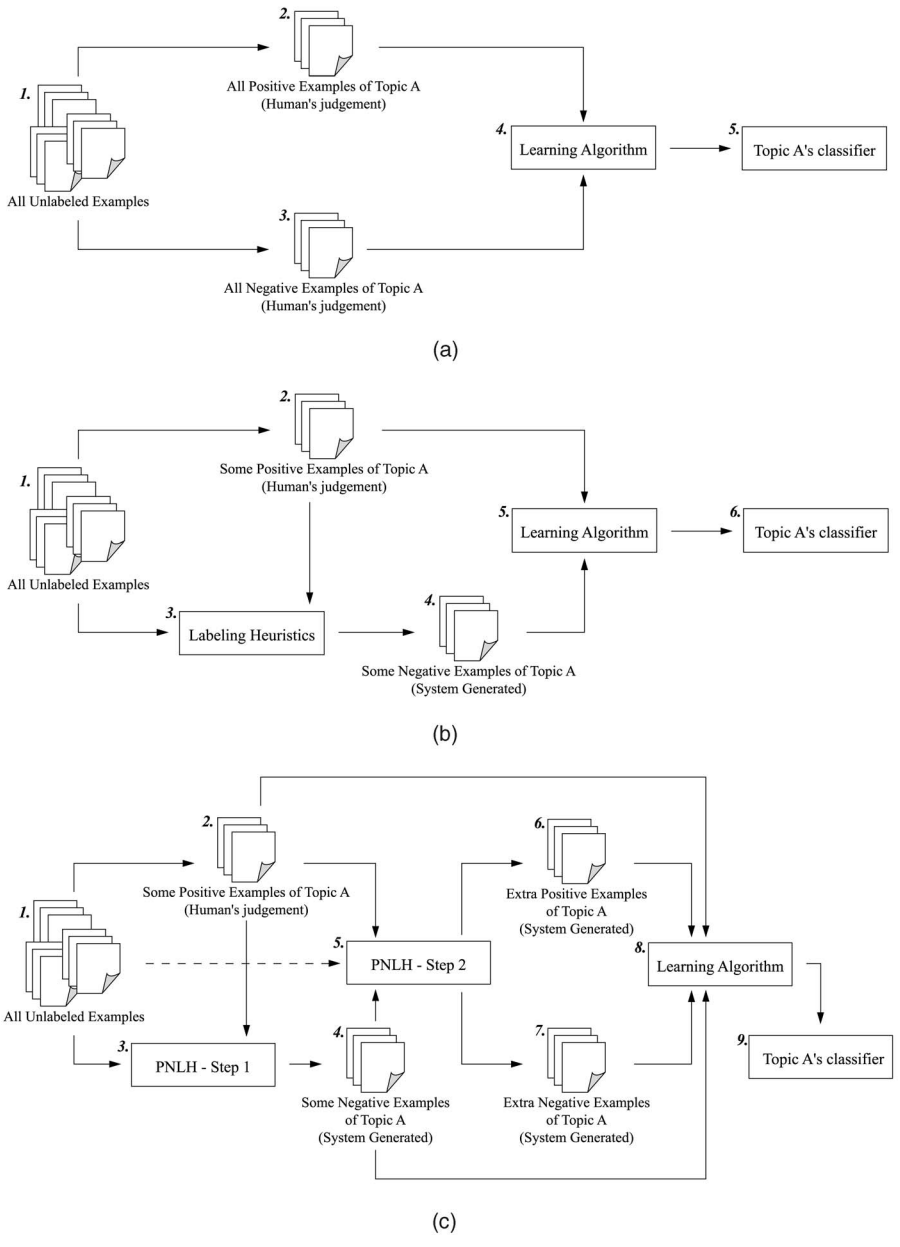

Fig. 1. A comparison of different learning frameworks.

step, the classifier is built[1] using the given positive example, P, and the extracted negative examples, N.

Surprisingly, none of the existing heuristics try to enlarge the given positive examples P by extracting the positive examples from U into P, with the one exception of our previous work [5].[2] Throughout our study, we found that extracting P' from U is very difficult. The reason is that simply comparing the differences of the feature distributions between P and U cannot help to extract P'. A document in U that possesses the features exhibited in P does not necessarily mean that it is a positive example. The very large size of and the very high diversity in U also contribute to the difficulties of extracting P'.

In this paper, we propose a labeling heuristic, called *PNLH* (**P**ositive examples and **N**egative examples **L**abeling **H**euristic) which is an extension of our preliminary work in [5]. Fig. 1c shows the general framework of *PNLH*. It consists of two steps: extraction and enlargement.

- **Step 1: Extraction**. This step aims at extracting a set of negative examples called *reliable negative examples* (N) from the unlabeled examples (U). The extraction is based on the concept of *core vocabulary*. This concept is first introduced in our previous work [5]. Core vocabulary contains the features with *feature strengths* greater than a threshold. However, in contrast to [5] where this threshold must be *predefined*, this paper proposed a new function to

1. Some of the papers (e.g., [9], [11], [13], [24]) modify the traditional classification algorithm so as to build the classifier *interactively*. Details will be discussed later.

2. Although [12] also builds a classifier by enlarging both the positive documents, our work is very different from that because our whole learning process is automatic, while [12] requires manual indexing.

compute the feature strength effectively such that the threshold can be determined *automatically*. With the help of the *core vocabulary of positive examples* (in the following, we use *positive features* for short), we then extract N from U.

Apart from the issues related to computing feature strengths effectively and determining a threshold for the core vocabulary automatically, this paper further addresses the question about whether two documents should be considered as positive examples, if both of them contain the *same number* of but *different* positive features. Based on our extensive experiments, we found that two documents that have the same number of positive features do not necessarily mean that both of them are positive examples. In this paper, we propose a set of novel functions to determine whether a document should be considered as a positive example based on ranking. Under these new proposals, we can increase the number of reliable negative examples and meanwhile maintain the quality of extraction.

- **Step 2: Enlargement**. This step aims at extracting a set of positive examples (P') and another set of negative examples (N') from U' ($U' = U - N$), so as to enlarge P by P' and N by N'. It consists of two steps:

  1. First, we partition the reliable negative examples (N) that are extracted in Step-1 into k clusters ($N_1, N_2, \cdots, N_k$). In contrast to [5] where the number of partitions (k) is *predefined* and is the *same* for all categories in the same domain, this paper presents techniques for determining k *dynamically* and *automatically* such that k may vary among categories even when the categories are within the same domain.
  2. Second, we extract P' and N' from U' so as to enlarge the P and N. In contrast to [5], where it only considers the similarity among P, N_i, and U' when performing the extraction, this paper further takes the number of features within the domain into consideration. We found that reducing the number of features in this enlargement step can significantly reduce mislabeling while still retaining as many positive and negative examples extracted as before. The reason why reducing the number of features can improve the quality of extraction is due to the fact that P and N_i are both very noisy.[3] Reducing the number of features in P and N_i can possibly diminish the noise and, therefore, improve the quality of extraction.

Finally, we use $P \cup P'$ as the positive training instances ($\mathcal{P}$) and $N \cup N'$ as the negative training instances ($\mathcal{N}$) to build the classifier. Note that applying *PNLH* in any domain is independent of the classifier built. We have conducted extensive experiments to address two issues: 1) how much *PNLH* can achieve on top of several representative classifiers and 2) how much *PNLH* can achieve, in comparison with other heuristics, on the same classifier. Our results indicated that our approach is highly feasible even when the set of positive examples is extremely small. We do not need to build any complex classifier interactively as those reported in [9], [11], [13], [24].

The rest of the paper is organized as follows: Section 2 presents *PNLH* in detail. Section 3 reviews the major heuristics that tackle the problem of partially supervised learning. Section 4 evaluates our work. We summarize and conclude this paper in Section 5.

2 PNLH: Positive Examples and Negative Examples Labeling Heuristics

The overview of the Positive examples and Negative examples Labeling Heuristic (*PNLH*) is shown in Fig. 1c and Algorithm 1. *PNLH* consists of two steps: Extraction and Enlargement. The objective of Extraction is to extract a set of *reliable negative examples* (N) from the unlabeled examples (U) (line 1 of Algorithm 1). The objective of Enlargement is to further extract positive examples (P') and negative examples (N') from $U - N$ (line 3 of Algorithm 1), so as to enlarge P and N.

Algorithm 1. *PNLH*(P, U).
Input: P (positive examples) and U (unlabeled examples)
Output: $\mathcal{P}$ (positive training examples) and $\mathcal{N}$ (negative training examples)
1. $N \leftarrow$ *ExtractReliableNegative*(P, U);
2. $U' \leftarrow U - N$;
3. obtain P' and N' by calling *Partition*(P, N, U');
4. $\mathcal{P} \leftarrow P \cup P'$;
5. $\mathcal{N} \leftarrow P \cup P'$;
6. **return** $\mathcal{P}$ and $\mathcal{N}$

2.1 Extracting Reliable Negative Examples

In the absence of any prior knowledge about the characteristics of the negative examples, the best way to extract a set of negative examples is to use the differences of the feature distributions between the given positive examples (P) and the unlabeled examples (U). In the following, we call the negative documents extracted from U *reliable negative examples* and denote them by N. We will show how reliable N is in the experimental studies in Section 4. There are two main procedures in this step, namely, identifying positive features and extracting reliable negative examples.

2.1.1 Identifying Positive Features

Positive features are the features that frequently appear in P and can represent P well. We identify the positive features based on a notion called *core vocabulary*. Note that a document that belongs to P must possess some of the features that are contained in the core vocabulary of P, denoted by V_P (Remark 2.1). According to V_P, we extract the reliable negative examples (N) from the unlabeled examples (U).

Remark 2.1. A document that belongs to the positive examples (P) must possess some of the features that are contained in the core vocabulary of P (V_P).

3. The definition of noisy will be discussed in Section 4.3

We have to stress that we are *not* trying to identify *all* negative examples in U simply using the concept of core vocabulary. We only aim at identifying some negative examples that are *reliable* in this step. In fact, it is impossible to extract all negative examples by simply comparing the feature distributions, as the feature distributions in the text domain are sparse [19].

The core vocabulary of P, V_P, is identified by comparing the *feature strengths* among the features that are exhibited in P. Without loss of generality, let $H(f_j)$ be a function to compute the feature strength of a particular feature f_j. A feature f_j is assigned to V_P, i.e., $f_j \in V_P$, if $H(f_j)$ is greater than a threshold θ. A feature with a higher feature strength indicates that it is more *effective* in terms of classification. In the following, we give the details of how to compute the function $H(f_j)$ and show how to identify the threshold θ. It is important to know that users do not need to specify such θ explicitly.

Remark 2.2. Let $H(f_j)$ be a function that computes the feature strength of a particular feature f_j, such that $H(f_a) > H(f_b)$ if and only if f_a is more effective (higher discriminative power) than f_b in terms of classification. Furthermore, $f_j \in V_P$ if and only if $H(f_j) > \theta$ for $\theta \in \Re$.

Techniques for approximating feature strength, $H(f_j)$, can be found from probabilistic theory, such as information gain, χ^2, odd ratio, and mutual information [20]. Unfortunately, *none* of them is suitable to solve our problem. The reason is that we do not have any prior knowledge about the feature distribution between the positive examples and negative examples over the entire document domain. Furthermore, it is impossible to obtain or estimate these values. As a result, obtaining the feature strengths is a very difficult and nontrivial task because we cannot use any of the existing methods to compute it.

In this paper, we approximate $H(f_j)$ by proposing a formula that can utilize the information presented in P and U. Let $n_P(f_j)$ and $n_U(f_j)$ denote the number of documents that contain f_j in P and U, respectively. $H(f_j)$ is computed by measuring the differences of the normalized document frequency between P and U:[4]

$$H(f_j) = \frac{n_P(f_j) - min_P}{max_P - min_P} - \frac{n_U(f_j) - min_U}{max_U - min_U}, \quad (1)$$

where max_P and min_P are the maximum value and minimum value of $n_P(f_j)$ for $f_j \in P$. A similar formula applies to max_U and min_U.

In (1), the first component and the second component correspond to the *normalized* number of documents that contain f_j in P and U, respectively. Although some of the existing works (e.g., [23], [24]) also compute $H(f_j)$ based on some newly-defined formulas, *none* of them pays attention to the significant of normalization. However, normalization is critical. This is because the differences between $|P|$ and $|U|$ are extremely large ($|P| \ll |U|$), such that only computing normalization in (1) makes it possible to compare the true relative frequencies of features between P and U.

Moreover, (1) is robust since it captures the idea that a feature is more important if its distribution is highly skewed toward P. To be more specific, if the normalized frequency of a feature in both P and U is similar, then its feature strength will approach to zero (i.e., $H(f_j) = 0$). If it is skewed toward one set, its feature strength will be either strictly positive (if its distribution is skewed to P) or strictly negative (if its distribution is skewed to U).

Recall that V_P is identified by extracting f_j such that $H(f_j) > \theta$. Here, we determine θ as the average of $H(f_j)$ for $f_j \in P$, i.e.:

$$\theta = \frac{1}{N_P} \sum_{f_j \in P} H(f_j), \quad (2)$$

where N_P is the number of different features in P.

2.1.2 Extracting Reliable Negative Examples

We have shown how to compute $H(f_j)$ and determine θ so as to select a set of positive features for formulating the core vocabulary of the positive examples (V_P). Another important issue is, given a document, how many positive features should this document have in order to consider it a potential positive example? In other words, what are the number of positive features that a reliable negative example in U should not exceed?

The choice of such a number is sensitive as it varies from different documents in U. There should not exist such a single number that applies to all documents in U and for all different categories. If we use a small number, then, although the precision of extracting N will be high, the recall would be very low, i.e., the size of N would be very small. This, in turn, makes the effort of this step ineffective. In contrast, if we use a large number, then the recall would be high, but the precision would be very low.

In sight of this, we propose a ranking-based approach for determining whether a document in U should be regarded as a reliable negative example. Consider two documents, d_i and d_j, such that both of them contain the same number of features in V_P. If the document d_i contains features with higher rank in V_P than d_j does, we say that d_i is more likely to be a positive example than d_j. Based on the ranking of the feature strengths using (1), we assign a unique value, $\nu(f_j)$, to every feature in V_P. We call $\nu(f_j)$ the *positive referencing power*. The positive referencing power is computed based on the standard exponential distribution [15]:

$$\nu(f_j) = \frac{1}{|V_P|} e^{-\frac{1}{|V_P|}r}, f_j \in V_P, \quad (3)$$

where r is the rank of feature f_j in V_P. Note that $0 < \nu(f_j) < 1$. Given a document, d_i, let $G(d_i)$ be the positive referencing power of d_i. It is computed by simply averaging all of the $\nu(f_j)$ in d_i, i.e.:

$$G(d_i) = \frac{1}{m} \sum_j \nu(f_j)) \text{ for all } f_j \in d_i, \quad (4)$$

where m is the number of positive features in d_i. A document, d_i, is considered a reliable negative example if

4. It is based on the observation that the number of positive examples is small when the diversity is high in a large U.

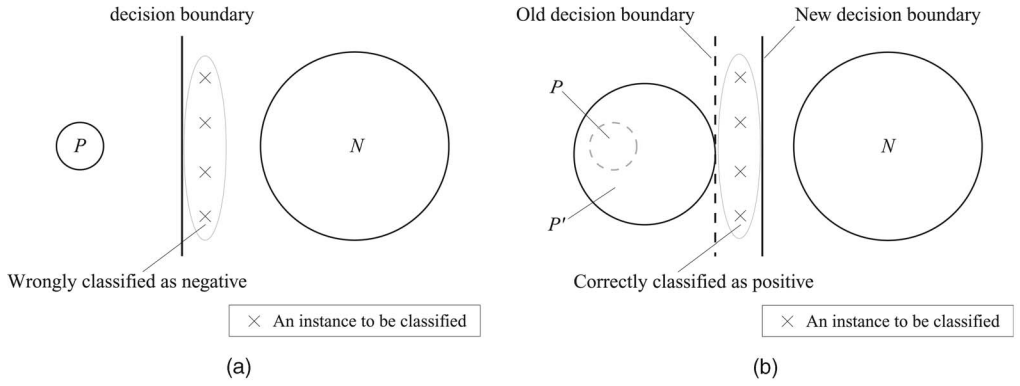

Fig. 2. When P is enlarged, the recall of the classifier would be increased. (a) Without enlarging P. (b) Enlarging P with P'.

$G(d_i) < \phi$. Here, ϕ is defined as the average of $G(d_i)$ for all documents in P:

$$\phi = \frac{1}{|P|}\sum_{i=0}^{|P|}(G(d_i)). \quad (5)$$

Remark 2.3. Using both $H(\cdot)$ and $G(\cdot)$ together will extract a larger but higher quality set of reliable negative examples than with $H(\cdot)$ alone. This is because $G(\cdot)$ allows some documents which contain some higher ranked features in V_P to be negative examples while excluding those documents which contains very many lower ranked features in V_P in the set of negative examples.

The *ExtractReliableNegative* algorithm, which aims at extracting N, is outlined in Algorithm 2. It takes two inputs, a small set of positive examples (P) and a large set of unlabeled documents (U). In Lines 1-3, it computes $H(f_i)$ (1) for all features, f_i, belonging to P. In Line 4, it computes the threshold θ using (2). The core vocabulary of P (V_P) is determined in Lines 5-10. In Line 11, it computes the threshold ϕ (5). Finally, the reliable negative examples are extracted in Lines 13-17, and the result is returned in Line 18.

Algorithm 2. *ExtractReliableNegative*(P, U).
Input: P (positive examples) and U (unlabeled examples);
Output: N (reliable negative examples);
1. **for all** $f_j \in P$ **do**
2. compute $H(f_j)$ using (1);
3. **end for**
4. compute θ using (2);
5. $V_P \leftarrow \emptyset$;
6. **for each** $f_j \in P$ **do**
7. **if** $H(f_j) > \theta$ **then**
8. $V_P \leftarrow V_P \cup \{f_j\}$;
9. **end if**
10. **end for**
11. compute ϕ using (5);
12. $N \leftarrow \emptyset$;
13. **for all** $d_i \in U$ **do**
14. **if** $G(d_i) < \phi$ **then**
15. $N \leftarrow N \cup \{d_i\}$;
16. **end if**
17. **end for**
18. return N;

2.2 Enlarging the Sets of Positive Examples and Negative Examples

In the previous section, we discussed how to extract the reliable negative examples (N) from the unlabeled examples (U) based on the positive examples (P). In this section, we present how to extract positive examples (P') and another set of negative examples (N') from U', where $U' = U - N$. By doing so, we can enlarge the given P by P' and the previously extracted N by N'. The enlargement is motivated by the following two observations:

- If $|P|$ is small, the number of positive examples for training is inadequate. This is because P cannot reflect the true feature distribution of all the positive examples in the domain. Enlarging the total number of positive examples will increase the recall of the classifier. Fig. 2 illustrates this idea. When P is enlarged, the situation where the positive examples are wrongly being classified as negative examples (Fig. 2a) can be possibly be corrected (Fig. 2b).
- If $|P|$ is large, $|V_P|$ will be large, and the number of reliable negative examples (N) that is extracted in the previous step will be small. Enlarging the total number of negative examples will increase the precision of the classifier. Fig. 3 illustrates this idea. When N is enlarged, the situation where the negative examples are wrongly being classified as positive examples (Fig. 3a) can possibly be corrected (Fig. 3b).

In text domains, the distribution of features are very sparse. Typically, the number of features can be ranged from 10,000 to 100,000, where the common features among documents are very few. Hence, even if we only enlarged P and N by few examples, the entire space occupied by the positive and negative examples may change significantly. In short, enlarging P and/or enlarging N properly will improve the quality of classification, as the boundary between the positive examples and negative examples will be changed dramatically.

Unfortunately, properly extracting P' and N' from U' is a very challenging task. The difficulty in extracting a proper set of P' from U' is caused by the fact that a document in U' that possesses some features in V_P may not necessarily be a positive example. In addition, the proportion of P' in U' is usually very small, which makes extracting P' even harder.

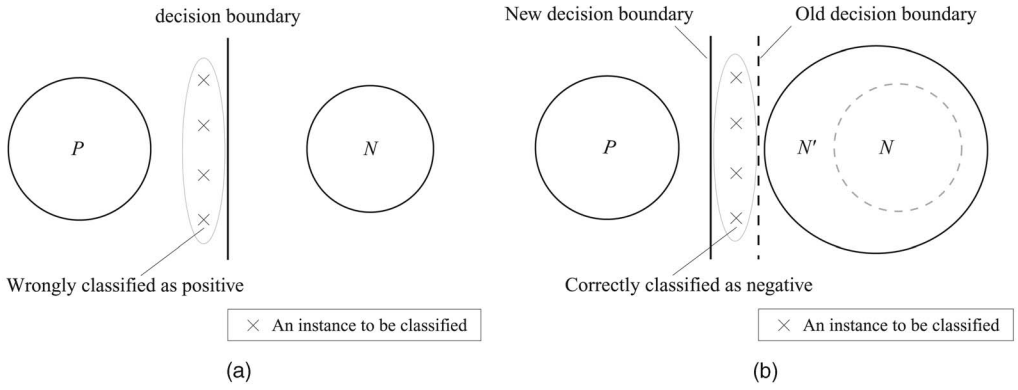

Fig. 3. When N is enlarged, the precision of the classifier will be increased. (a) Without enlarging N. (b) Enlarging N with N'.

We will discuss this issue in the following sections in more detail. In the following sections, we first discuss some *possible* but *inappropriate* approaches to extract P'. Then, we present our solution which is based on a partition strategy.

2.2.1 Possible but Inappropriate Approaches

In this section, we discuss three intuitively possible but inappropriate approaches for extracting P' and N'. We summarize them below and discuss them in detail in the following sections.

- **Reverse Extraction Approach (RE)**: This approach adopts the same idea as described in the previous section, but implements in a reverse direction. It first constructs a core vocabulary for the reliable negative examples, V_N, from N, and then extracts P' from U' such that none of the documents in P' possesses the features that are listed in V_N.
- **Simple Text Classifier Construction Approach (TC)**: This approach first builds a traditional text classifier based on P and N, and then classifies U'. The documents that are classified as positive are regarded as P', and the documents that are classified as negative are regarded as N'. This approach is reported in [11], [10], [13], [24] for extracting N', but not P' as we claim that it cannot extract a high quality of P'.
- **Profile-Based Approach (PB)**: This approach first constructs a profile for each class, P and N. Given a document, d, in U', in order to classify whether d belongs to P' or N', it compares the distance between d and the profile of P against the distance between d and the profile of N. This technique is adopted in [10], [11] for extracting N', but not P'.

2.2.2 Reverse Extraction Approach (RE)

RE is inappropriate for extracting both P' and N'. Note that U consists of many different topics (categories) and does not focus on one single topic. In most text classification problems, the level of diversity is always very high. Since $N \subset U$, N inherits this property. As every topic in N contains its own set of core vocabulary, a large number of different topics in N cancels the significance of each others' core vocabulary. Eventually, none of the features in N can be properly selected or the features selected are of poor quality.

2.2.3 Simple Text Classifier Construction Approach (TC)

TC can possibly extract a high quality of N', but is impossible to extract a high quality of P'. Our argument is based on our observations on two major kinds of classifiers: *kernel-based classifier*, like SVM and regression models, and *instance-based classifier*, like kNN.

- **Kernel-Based Classifier**. Kernel-based approaches may wrongly classify many negative examples into P'. The reasons are as follows: Recall that N consists of diverse topics and covers a very large region in the feature space. On the other hand, P only focuses on one single topic and covers a much smaller region in the same feature space. When $|P|$ is small, P may be inadequate for reflecting the true feature distribution of all positive instances in the domain. To be more specific, let us take SVM as an example. Figs. 4a and 4b show how SVM constructs its decision function by maximizing the boundary between positive training instances and negative instances and minimizing the errors. In both figures, N (the crosses) are the same, but P (the black dots) are different. Note that $|P|$ in Fig. 4a is smaller than that in Fig. 4b. Let us refer to Fig. 4a first. When the number of positive training instances is inadequate, SVM may generate a wrong decision function (boundary) and cannot adjust it to the correct one, as shown in Fig. 4b. This explains why a kernel-based classifier always extracts P' with high recall but very low precision, whereas it extracts N' with high precision but low recall.
- **Instance-Based Classifier**. Although instance-based classifiers, like kNN [21], [22], do not relay on statistical distribution of the training data, they cannot extract a good set of positive examples, P'. Recall that a document containing some features that are listed in V_P cannot imply that it belongs to P. According to the probabilistic nature of feature distributions, any two documents may share a high degree of similarity even though they belong to different categories. Two documents can often be the nearest neighbor but do not belong to the same category. Table 1 shows the percentage of documents that their k nearest neighbors belong to different categories using three benchmarks as reference.[5]

5. We will discuss these benchmarks in Section 4. We use the cosine measure for computing their similarity.

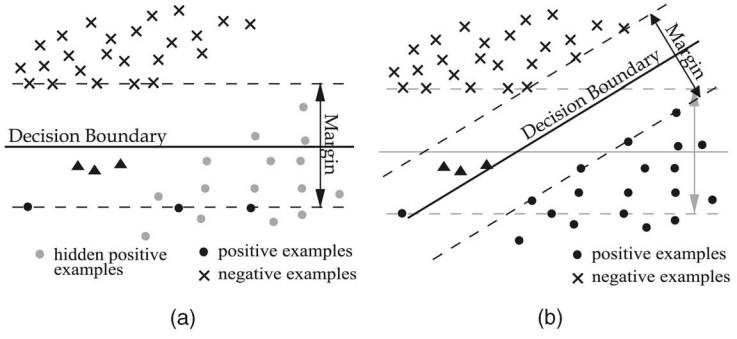

Fig. 4. The decision boundaries of SVM. (a) Small P (Inadequate). (b) Large P (Adequate)

Although the percentage may vary from data set to data set, it does confirm the aforementioned nearest neighbor behavior. This is why the threshold k in kNN will never be set to a small value in text classification. Typically, k is around 30-50 [7], [21], [22]. In summary, in order to construct a kNN classifier, we have first to define a proper k, and then tune a proper threshold ϕ as the decision function of the classifier. Unfortunately, these two issues are both difficult to answer in partially supervised learning problem. First, a large k is not good in our problem setting because $|P|$ is always small. A small k may obtain a poor result for kNN. Second, if $|P|$ is small, obtaining a validation data set is impossible. Hence, tuning a proper ϕ is very difficult as overfitting will occur.

2.2.4 Profile-Based Approach (PB)

PB, such as Centroid-based [20] or Rocchio [17], can possibly solve the problem of kNN. However, PB is still inappropriate for extracting a high quality of P'. To formalize our arguments, let us consider two profiles, C_P and C_N, which denote the documents in P and N, respectively. Suppose that we use cosine coefficient for measuring the similarity between a document, d, and a profile, C_i for $i \in \{P, N\}$:

$$S(d, C_i) = \frac{d \cdot C_i}{\| d \| \cdot \| C_i \|} \quad \text{for } i \in \{P, N\}. \tag{6}$$

Note that, for most applications, d will be normalized to unit length so as to account for the length of different documents [20], i.e., $\| d \| = 1$. For simplicity, let us use the centroids of the positive examples and negative examples to denote their corresponding profiles, i.e.:

$$C_P = \frac{1}{| P |} \sum_{d_i \in P} d_i, \qquad C_N = \frac{1}{| N |} \sum_{d_i \in N} d_i. \tag{7}$$

TABLE 1
Percentage of Documents whose First Five Nearest Neighbors Belong to Different Categories

NN	R21578 (10)	R21578 (90)	Newsgroup-20	Web-KB
First	11.9%	20.1%	30.6%	39.5%
Second	14.8%	25.6%	47.1%	43.5%
Third	17.4%	28.7%	56.5%	45.7%
Forth	18.4%	30.3%	62.6%	45.6%
Fifth	19.7%	31.8%	66.0%	47.7%

Let us consider $S(d, C_N)$; its numerator (6) now becomes:

$$d \cdot C_N = \frac{1}{| N |} \sum_{d_i \in N} S(d, d_i). \tag{8}$$

In short, the similarity between d and C_N, $S(d, C_N)$, is nothing more than the average similarity between d and *all* documents in N divided by $\|C_N\|$ ($\| d \| = 1$). As N is large and consists of many diverse topics, obviously, d can never be similar to *all* documents in N. Thus, $S(d, C_N)$ will be extremely small. This explains why the precision of P' extracted from U' is very low. The same observation is applied to other coefficients for computing similarity, such as the Jaccard coefficient or the Dice coefficient.

Similarly, for Rocchio, it also suffers from the above difficulties. In conclusion, it is difficult to extract P' from U' by building any kind of classifier simply using P and N directly.

2.2.5 A Feasible Partition-Based Approach

In this section, we present the second step of *PNLH*: A partition-based approach to extract P' and N' from U'. Recall that N consists of diverse topics. It is this diversity that makes all of the approaches that we discussed so far impractical to extract P' and/or N' from U'. The key issue is therefore finding a way to deal with the problem of diversity. In the following, we will discuss how to solve this problem using three main procedures, namely, partitioning the reliable negative examples, assigning documents to P' and N', and iterative enlargement.

2.2.6 Partitioning the Reliable Negative Examples

The partition-based approach further extends the idea of the profile-based approach by partitioning N into k partitions, $N_1, N_2, \cdots, N_k$, such that the documents in each partition share a high degree of similarity. By doing so, each partition focuses on a smaller set of more related features. In order to extract P' and N' from U', we compare the similarity of the documents in U' with the centroid of P (C_P) and the centroid of N_i (C_{N_i}) for $i = 1, 2, \cdots, k$.

For partitioning the reliable negative example, N, we can apply any kind of the existing clustering algorithm. In the paper, for simplicity, we adopt the classical k-means clustering algorithm [3], [4], [8]. A common question regarding clustering is: What is the best number of clusters, k? If k is too large, the problem that rises in the

instance-based approach appears. If k is too small, the problem that rises in the profile-based approach appears.

Suppose that an optimal number of partitions, k, does exist. This optimal number should vary from one domain to others. Furthermore, even in the same domain, different categories should have different optimal values.

In this paper, we determine k based on the size of the given positive examples (P) and the reliable negative examples (N) such that $k = \sqrt{|N|/|P|}$. The idea behind this is that k needs to be controlled in a way as to maintain a reasonable number of documents in clusters N_i. If k is large as $|N|/|P|$, the number of documents in some cluster N_i may be too small. Such a value is confirmed based on our extensive testing results. Some observations are given below. Consider two categories, x and y. x and y have P_x and P_y positive examples and N_x and N_y extracted reliable negative examples, respectively. Suppose $|P_x| \gg |P_y|$. Then, the set of extracted reliable negative examples, N_x, is much smaller than N_y, i.e., $|N_x| < |N_y|$. Specifically, if $|P|$ is small, then V_P will be small, which, in turn, makes N_i become larger. Similarly, if $|P|$ is large, then V_P will be large, which, in turn, makes $|N_i|$ become smaller. Consequently, the diversity in N_x is smaller than that of N_y. Therefore, we should set a small k to partition N_x and a large k to partition N_y. It leads to the following remark: If $|P|$ is large and $|N|$ is small, then k should be small, whereas if $|P|$ is small and $|N|$ is large, then k should be large. The above equation reflects the relationship among $|P|$, $|N|$, and k, and is shown to be very effective in our performance studies.

2.2.7 Assigning Documents to P' and N'

Without loss of generality, suppose we have partitioned the set of reliable negative examples, N, into k clusters, N_1, N_2, $\cdots, N_k$, as discussed above using a clustering algorithm. In this section, we show how to assign documents from U' to P' as in N', so as to enlarge the total positive training instances $\mathcal{P}$ and the total negative training instances $\mathcal{N}$ (Fig. 1c). Before we continue our discussion, let us stress that, if a document is assigned to N_i, the document will be assigned to N'.

The assignment of a document $d \in U'$ to either P' or N' cannot be done by simply comparing the similarities between d and C_P and between d and C_{N_i} using (6). Recall that C_P and C_{N_i} are centroids for P and N_i. In other words, we cannot say that d is a positive example if $S(d, C_P)$ is less than the closest of $S(d, C_{N_i})$ for any cluster N_i using (6), and we cannot say that d is a negative example if $S(d, C_P)$ is greater than the closest of $S(d, C_{N_i})$ for any cluster N_i using (6). The reason is that, in terms of similarity measurement, d may be similar to both P' and some N_i. What we need is to extract the document d which is *significantly* similar to either P or some N_i. Documents which are similar to both P and some N_i are regarded as ambiguous and will be ignored.

We propose the following strategy: A document d in U' belongs to P' if it satisfies the following conditions:

$$S(d, C_P) > \mu_P \tag{9}$$

$$M_P(d) = S(d, C_P) - \max S(d, C_{N_i}) > \gamma_P, \tag{10}$$

where μ_P is the average similarity of the documents within P and γ_P is the average difference between the documents in P and the closest N_i ($\subseteq N$):

$$\mu_P = \frac{1}{|P|} \sum_{d_i \in P} S(d_i, C_P) \tag{11}$$

$$\gamma_P = \frac{1}{|P|} \sum_{d_i \in P} \Big(S(d_i, C_P) - \max_{j=1,\cdots,k} S(d_i, C_{N_j}) \Big). \tag{12}$$

Note that, from the above equations, $S(d, C_P) > \mu_P$ can guarantee d is similar to P, while $M_P(d) > \gamma_P$ can ensure d is not similar to both P and N. In a similar fashion, a document d in U belongs to N' if and only if the following conditions hold:

$$S(d, C_{N_i}) > \mu_{N_i}, \text{ for some } i \tag{13}$$

$$M_{N_i}(d) = S(d, C_{N_i}) - S(d, C_P) > \gamma_N, \text{ for some } i, \tag{14}$$

where

$$\mu_{N_i} = \frac{1}{|N_i|} \sum_{d_i \in N_i} S(d_i, C_{N_i}) \tag{15}$$

$$\gamma_N = \frac{1}{k} \sum_{i=0}^{k} \frac{1}{|N_i|} \sum_{d_j \in N_i} \Big(S(d_j, C_{N_i}) - S(d_j, C_P) \Big). \tag{16}$$

The above strategy works well when the size of the positive examples (P) is of reasonable size. However, the set of positive examples may vary greatly:

- If $|P|$ is very small (say, around 10), then the similarity among the documents in P will be very high. Consider an extreme case. If $|P| = 1$, then the similarity will be 1, which implies μ_P will be 1 and γ_P will be very large. As a result, no documents in U' will be selected as positive. The same observations can be made for N_i.
- If $|P|$ is large, then P will contain many different features because the distribution of features are always sparse in text domains. The chance that a document processes some positive features but not a positive example increases. Moreover, the similarity within P will be very low, which, in turn, makes both μ_P and γ_P very small. Many documents in U' will be very similar to P and/or N_i. Eventually, ambiguity appears. The same observations can be made for N_i.

Mathematically, following (7)-(6), we have the following:

$$S(d, C_P) = \frac{1}{||C_P||} \cdot \frac{1}{|P|} \sum_{d_j \in P} S(d, d_j) \tag{17}$$

$$S(d, C_{N_i}) = \frac{1}{||C_{N_i}||} \cdot \frac{1}{|N_i|} \sum_{d_j \in N_i} S(d, d_j). \tag{18}$$

It is important to notice that both $|N_i|$ and $||C_{N_i}||$ in the above equations are dominate factors as the features are always sparse in nature. As a result, the summation always has a *small value*. Thus, the comparison between d and C_P as well as between d and C_{N_i} is most likely to be affected by the size of each feature set, rather than the features exhibited in P and N_i.

The above suggests that we cannot follow (7) to construct the centroids of C_P and C_{N_i} by simply averaging the weights of the features in the corresponding P or N_i. In this

paper, we propose an efficient and effective approach to construct the centroids, C_P and C_{N_i}, by only using a set of representative features in the corresponding centroid. Consequently, these centroids will be more dense, and this makes comparison of similarity between a document d and the centroid C_P or between a document d and the centroid of C_{N_i} more efficient. Our strategy is given in the following remark.

Algorithm 3 shows the steps for constructing C_P and C_{N_i}. First, in Line 1, it selects the top n representative features from $P \cup N$. In this work, we set $n = 3,000$ based on a series of testing for selecting features in the range from $1,000$ to $5,000$. More details about the issues of feature selection would be discussed in our experimental studies in Section 4.3. The centroid C_P is those top n features that exhibit in P (Lines 3-6). The centroid C_{N_i} is those top n features that do not belong to C_P but belong to N_i (Lines 7-15).

Algorithm 3. *Centroid*(P, $N_0, N_1, \ldots, N_k$).
Input: P (positive documents), $N_0, N_1, \ldots, N_k$ (negative documents in different partitions);
Output: C_P (centroid of the positive class) and $C_{N_0}, C_{N_1}, \ldots, C_{N_k}$ (negative centroids in different partitions);
1. $L \leftarrow$ top n features ranked by information gain from $P \cup N$;
2. $C_P \leftarrow \emptyset$;
3. **for each** $f_i \in P$ **do**
4. $\quad C_P \leftarrow C_P \cup \{f_i\}$
5. **end for**
6. construct the weights of C_P by averaging the weights of the features in P;
7. **for all** $N_i, i \in \{1..k\}$ **do**
8. $\quad C_{N_i} \leftarrow \emptyset$;
9. $\quad$ **for each** $f_j \in L$ **do**
10. $\quad\quad$ **if** $f_j \in N_i$ **then**
11. $\quad\quad\quad C_{N_i} \leftarrow C_{N_i} \cup \{f_j\}$;
12. $\quad\quad$ **end if**
13. $\quad$ **end for**
14. $\quad$ construct the weights of C_{N_i} by averaging the weights of the features in N;
15. **end for**
16. **return** C_P and $C_{N_0}, C_{N_1}, \ldots, C_{N_k}$;

2.2.8 Iterative Enlargement

The whole enlargement process for extracting P' and N' from U' is outlined in the *Partition* algorithm (Algorithm 4). This algorithm repeatedly extracts negative examples from U' until no more documents in U' (Lines 2-13) can be extracted. After it has extracted all N', it begins to extract P' from U' (Lines 15-19). We can obtain a set of positive training instances ($\mathcal{P}$) by merging the given P and the extracted P', and a set of negative training instances ($\mathcal{N}$) by merging the two sets of extracted negative documents, $N \cup N'$. According to $\mathcal{P}$ and $\mathcal{N}$, a classifier can be built.

Algorithm 4. *Partition*(P, N, U').
Input: P (positive documents), N (negative documents), and U (unlabeled documents);
Output: P' (positive documents) and N' (negative documents);
1. $P' \leftarrow \emptyset$; $N' \leftarrow \emptyset$;
2. **repeat**
3. $\quad$ partition $N \cup N'$ into k clusters, $N_0, N_1, \ldots, N_k$;
4. $\quad$ obtain C_P and $C_{N_0}, C_{N_1}, \ldots, C_{N_k}$, by calling *Centroid*($P, N_0, N_1, \ldots, N_k$);
5. $\quad U' \leftarrow U - N'$;
6. $\quad i = 0$;
7. $\quad$ **for all** $d \in U'$ **do**
8. $\quad\quad$ **if** d does not satisfy the conditions in (9) and (10) **and** d satisfies the conditions in (13) and (14) **then**
9. $\quad\quad\quad N' \leftarrow \{d\} \cup N'$;
10. $\quad\quad\quad$ i++;
11. $\quad\quad$ **end if**
12. $\quad$ **end for**
13. **until** $i = 0$
14. $U' \leftarrow U - N'$;
15. **for all** $d \in U'$ **do**
16. $\quad$ **if** d satisfies the conditions in (9) and (10) **and** d does not satisfy the conditions in (13) and (14) **then**
17. $\quad\quad P' \leftarrow \{d\} \cup P'$;
18. $\quad$ **end if**
19. **end for**
20. **return** P' and N';

3 Related Work

This section briefly discusses the existing works that are related to the extraction and enlargement of negative examples. There are five major approaches for extracting N from U given P, namely, Rocchio [10], [11], Spy [13], PEBL [25], [24], Weighted-Logistic Regression [9], and Naive Bayes [12].

- **Rocchio**: This labeling heuristic is proposed in [10], [11]. In order to extract N from U, it first builds a traditional Rocchio classifier [17] using the given P as the positive training example and U as the negative training example. After that, it classifies the documents in U using this Rocchio classifier. The documents that are classified as negative form the negative training examples $\mathcal{N}$. In [11], Liu et al. also show that using SVM iteratively to extract a set of negative examples, N' from $U' = U - N$, can possibly improve the effectiveness of the classifier.
- **Spy**: Spy is proposed in [13]. It first randomly selects a set of documents, $S \subset P$, and puts them into U. The default size of S is 15 percent of P. Here, S serves as *spy documents* in U. It assumes that the spy documents behave as similar as the unknown positive examples in U, hoping to infer the behavior of these unknown positive examples. It then runs an I-EM (iterative-EM) algorithm using the set of S' ($S' = P - S$). After I-EM completes, the resulting classifier uses a probabilistic approach based on S to decide a threshold to identify N. In [11], Liu et al. show that further extracting N' from U' using SVM iteratively cannot improve the effectiveness of the classifier.
- **PEBL**: PEBL is proposed in [25], [24]. It first builds a feature set, F, which contains features, f_j, that occur in P more frequently than in U. Any document in U

that does not contain the features in F is classified as N. In [11], Liu et al. show that further extracting N' from U' using SVM iteratively can improve the effectiveness of the classifier when $|P|$ is small but deteriorate when $|P|$ is large. However, in some situations, iterative-SVM will improve the effectiveness of the classifier when $|P|$ is large but deteriorate when $|P|$ is small. Heterogeneous Learner (HL) [23] uses a similar but more sophisticated approach for Web page classification. However, we do not include it in the experimental studies as we cannot get a satisfactory result if we use HL.

- **Weighted Logistic Regression**: This heuristic is proposed in [9]. It uses a logistic regression on the weighted examples together with a performance measure that is estimated from both the labeled positive documents and the unlabeled documents so as to select a regularization parameter for a validation set. The authors claim that this heuristic can be applied in any data set, not just text data.
- **Naive Bayes**: This heuristic is proposed in [12]. Instead of labeling the documents, this heuristic labels a set of representative words for each category by *human experts*. It then uses these words to extract a set of documents for each category from a set of unlabeled documents to form the initial training set. EM is then applied to build the final classifier. Since this heuristic requires significant human indexing and the evaluation is subjective, we do not include it in our experimental studies.

4 Experimental Studies

Extensive experiments are conducted to verify the effectiveness of *PNLH*. For the document preprocessing, punctuation, numbers, Web page addresses, and e-mail addresses are removed. All features are stemmed and converted to lower cases. Features that only appear in one document are ignored. All features are weighted using the traditional $tf \cdot idf$ schema [18] and normalized to unit length [20]. Different versions of the $tf \cdot idf$ schema are implemented and their influences are insignificant or none. Due to the space limited, for evaluating the effectiveness of the classifier, we only report the micro-F1 value [20], which is a balance between the precision and recall. Unless otherwise specified, we used the following settings by default:

1. For SVM, we used linear kernel with $C = 1.0$.
2. For Naive Bayes, we use multinomial version.
3. For Rocchio, we turn the values of α and β to its optimal values.
4. Feature selection is applied to Naive Bayes and Rocchio, and we report the best F1 score.

4.1 Experiment Setup

The following benchmarks are used:

- **Reuters-21578**: Following the recent trend, we use the ModApte split to separate the data into a training set and a evaluation set. Note that Reuters-21578 is *highly skewed*. The largest category contains 2,877 training examples, while the smallest category contains one training example. Thus, we decide to have the following settings for this benchmark:
  - **Reuters-21578 (10)**: We select the top 10 largest categories for training and evaluation. This is one of the most popular ways for text classification evaluation [14], [7].
  - **Reuters-21578 (90)**: We use all of the 90 categories for training and evaluation. This is the standard method used in most text mining evaluation tasks.
- **Newsgroup-20**: There is a total of 20 different categories. Each category contains about 1,000 messages. Note that the number of messages in each category is nearly the same. There are 19,126 messages assigned to a single category and 320 messages assigned to multiple categories. For each category, we randomly select 80 percent of the messages as training data and the remaining 20 percent as testing data.
- **Web-KB**: There are 8,282 Web pages and seven categories. Each Web page belongs to one and only one category. Web-KB is slightly skewed. The largest category contains 3,764 Web pages, whereas the smallest category contains 137 Web pages. For each category, we randomly select 80 percent of the Web pages as training data and the remaining 20 percent as testing data.

For each benchmark, we conduct the following experiments: For each category k in the target benchmark, we randomly pick out x percent of the documents that belong to the category k and use these documents to form the positive examples, P. The remaining training documents in the benchmark are regarded as unlabeled examples, U. The experiment is repeated n number of times, and we report the average of the results. Here, n is 30 and x is ranged from $1, 2, \ldots, 9$ as small cases, and from $10, 20, \ldots, 100$ as large cases. Note that, when $x = 100$ percent, all positive examples are used, which is the benchmark case that the benchmark is designed for. In total, there are 19 cases (nine small cases, nine large cases, and one benchmark case).

4.2 PNLH versus without PNLH

In this section, we evaluate the effectiveness of *PNLH* in labeling the unlabeled training examples using three popular classifiers: SVM, Rocchio, and Naive Bayes (NB). Given a set of positive examples (P) and a set of unlabeled documents (U), for each classifier, we compare two cases: using *PNLH* versus not using *PNLH*. For the former, the positive examples and negative examples to each of the three classifiers are $\mathcal{P}$ and $\mathcal{N}$. For the later, the positive examples and negative examples to each of the three classifiers are P and U.

Tables 2, 3, 4, and 5 show the results using these benchmarks. In each table, the first column gives the number of percentage of the total number of positive examples used. The second three columns show the accuracy of the three classifiers when *PNLH* is not used, and the last three columns show the accuracy of the three classifiers when *PNLH* is used. As shown in these tables, the classifiers that use *PNLH* for labeling the training examples show higher accuracy, in particular, when the number of positive training examples becomes fewer.

All three classifiers with *PNLH* outperform the cases without *PNLH* up to $x = 50$ percent. The only exception is in Table 5 with Newsgroup-20; when $x = 1$ percent, Rocchio obtained 0.001 without *PNLH*, in comparison to 0 with *PNLH*.

TABLE 2
Results of Reuters-21578 (10)

%	Without *PNLH*			With *PNLH*		
	SVM	Rocchio	NB	SVM	Rocchio	NB
1	0.000	0.000	0.001	**0.483**	**0.469**	**0.442**
2	0.000	0.000	0.011	**0.501**	**0.469**	**0.538**
3	0.000	0.000	0.014	**0.588**	**0.651**	**0.552**
4	0.000	0.000	0.021	**0.625**	**0.698**	**0.611**
5	0.000	0.000	0.023	**0.677**	**0.686**	**0.631**
6	0.000	0.000	0.043	**0.692**	**0.706**	**0.646**
7	0.000	0.109	0.059	**0.716**	**0.713**	**0.668**
8	0.000	0.109	0.069	**0.716**	**0.734**	**0.693**
9	0.000	0.106	0.071	**0.727**	**0.757**	**0.700**
10	0.009	0.114	0.074	**0.729**	**0.754**	**0.704**
20	0.012	0.279	0.568	**0.757**	**0.770**	**0.752**
30	0.080	0.716	0.678	**0.789**	**0.804**	**0.780**
40	0.340	0.773	0.775	**0.804**	**0.825**	**0.799**
50	0.568	0.845	0.806	**0.824**	**0.846**	**0.816**
60	0.737	**0.866**	**0.831**	**0.824**	0.860	**0.831**
70	0.824	**0.873**	**0.843**	**0.862**	0.864	0.827
80	0.865	**0.873**	**0.841**	**0.874**	0.869	0.834
90	0.865	**0.876**	**0.845**	**0.887**	0.870	0.834
100	**0.907**	**0.872**	**0.846**	0.889	**0.872**	0.836

TABLE 3
Results of Reuters-21578 (90)

%	Without *PNLH*			With *PNLH*		
	SVM	Rocchio	NB	SVM	Rocchio	NB
1	0.000	0.002	0.002	**0.375**	**0.333**	**0.395**
2	0.000	0.020	0.007	**0.432**	**0.424**	**0.445**
3	0.000	0.015	0.011	**0.506**	**0.464**	**0.465**
4	0.000	0.023	0.014	**0.542**	**0.505**	**0.483**
5	0.000	0.026	0.012	**0.553**	**0.524**	**0.535**
6	0.000	0.038	0.021	**0.570**	**0.588**	**0.583**
7	0.000	0.046	0.017	**0.592**	**0.603**	**0.592**
8	0.000	0.195	0.104	**0.595**	**0.612**	**0.615**
9	0.000	0.105	0.327	**0.604**	**0.623**	**0.612**
10	0.006	0.111	0.044	**0.623**	**0.656**	**0.634**
20	0.009	0.477	0.513	**0.683**	**0.702**	**0.623**
30	0.035	0.595	0.602	**0.710**	**0.701**	**0.665**
40	0.270	0.764	0.651	**0.733**	**0.713**	**0.684**
50	0.468	0.797	0.702	**0.761**	**0.742**	**0.703**
60	0.631	**0.818**	**0.739**	**0.796**	0.783	0.715
70	0.730	**0.830**	**0.757**	**0.792**	0.792	0.712
80	0.791	**0.843**	**0.776**	**0.828**	0.814	0.724
90	**0.830**	**0.850**	**0.774**	0.808	0.814	0.744
100	**0.860**	**0.853**	**0.783**	0.819	0.826	0.754

TABLE 4
Results of WebKB

%	Without *PNLH*			With *PNLH*		
	SVM	Rocchio	NB	SVM	Rocchio	NB
1	0.000	0.003	0.000	**0.026**	**0.049**	**0.004**
2	0.000	0.008	0.003	**0.035**	**0.122**	**0.057**
3	0.000	0.018	0.011	**0.058**	**0.110**	**0.039**
4	0.000	0.021	0.014	**0.119**	**0.176**	**0.175**
5	0.000	0.030	0.018	**0.143**	**0.201**	**0.195**
6	0.000	0.043	0.021	**0.348**	**0.399**	**0.345**
7	0.000	0.077	0.019	**0.353**	**0.396**	**0.342**
8	0.000	0.087	0.025	**0.359**	**0.471**	**0.417**
9	0.000	0.103	0.036	**0.404**	**0.467**	**0.418**
10	0.005	0.075	0.065	**0.374**	**0.474**	**0.449**
20	0.029	0.169	0.132	**0.497**	**0.543**	**0.546**
30	0.062	0.173	0.275	**0.539**	**0.606**	**0.585**
40	0.151	0.280	0.415	**0.594**	**0.639**	**0.619**
50	0.260	0.401	0.485	**0.647**	**0.637**	**0.634**
60	0.364	0.522	0.564	**0.675**	**0.649**	**0.629**
70	0.475	0.612	0.615	**0.689**	**0.664**	**0.630**
80	0.544	0.641	**0.650**	**0.700**	**0.644**	0.629
90	0.664	0.666	**0.642**	**0.714**	**0.667**	0.640
100	0.717	**0.664**	**0.660**	**0.724**	**0.664**	0.645

TABLE 5
Results of Newsgroup-20

%	Without *PNLH*			With *PNLH*		
	SVM	Rocchio	NB	SVM	Rocchio	NB
1	0.000	**0.001**	0.000	0.000	0.000	0.000
2	0.000	0.007	0.001	**0.026**	**0.022**	**0.013**
3	0.000	0.016	0.002	**0.040**	**0.032**	**0.019**
4	0.000	0.034	0.005	**0.069**	**0.064**	**0.032**
5	0.000	0.036	0.010	**0.094**	**0.105**	**0.072**
6	0.000	0.055	0.021	**0.185**	**0.182**	**0.141**
7	0.000	0.044	0.024	**0.237**	**0.205**	**0.215**
8	0.000	0.049	0.034	**0.274**	**0.254**	**0.252**
9	0.000	0.069	0.046	**0.314**	**0.305**	**0.272**
10	0.001	0.103	0.070	**0.343**	**0.405**	**0.352**
20	0.013	0.290	0.185	**0.423**	**0.451**	**0.385**
30	0.029	0.365	0.299	**0.444**	**0.461**	**0.401**
40	0.073	0.514	**0.455**	**0.437**	**0.525**	0.452
50	0.162	0.618	0.532	**0.450**	**0.615**	**0.544**
60	0.301	0.659	0.610	**0.481**	**0.665**	**0.613**
70	0.438	**0.691**	**0.643**	**0.535**	0.684	0.623
80	0.573	**0.702**	**0.671**	**0.632**	0.684	0.642
90	**0.671**	**0.711**	**0.703**	0.662	0.695	0.654
100	**0.761**	**0.711**	**0.724**	0.722	0.701	0.695

Note that the capability of Rocchio and NB to tolerant noise[6] is much better than SVM. In general, Rocchio performs acceptably in the large cases ($\geq$ 40 percent). SVM's accuracy deteriorates significantly even if the number of positive examples decreases slightly.

It is interesting to see that, with WebKB (Table 4), when $x = 100$ percent (all positive examples), the accuracy obtained by SVM and Rocchio with *PNLH* perform even better, which indicates that *PNLH* can possibly further improve the classification results obtained by the traditional classification. In traditional classification, the documents that do not belong to a particular category are labeled as negative training examples. However, documents that do not belong to a particular category do not necessary fall into the negative training set. The content of these documents may be similar to the positive training examples. If we simply put these documents into the negative training set, the effectiveness of the classifier will possibly be affected.

When $|P|$ is very large, the effectiveness of the classifiers that do not use *PNLH* perform only marginally well. *PNLH* can boost the effectiveness of the classifiers and seldom harms it.

4.3 The Reliability of the Reliable Negative Examples (Step 1)

Recall that Step 1 of *PNLH* is to extract a set of reliable negative examples. To answer how reliable the extracted reliable negative examples are, we use two measures, *precision* and *purity*. By purity, we mean what percentage of the positive examples are mixed with the reliable negative examples. Let P be the whole set of positive examples. Note: P is the set of x percent sampled positive examples and N is the set of extracted negative examples:

$$purity = 1 - \frac{|\mathbf{P} \cap N|}{|\mathbf{P}|}. \quad (19)$$

6. In general, there are two kinds of noises: noises in the feature and noises in the training data. The former one is usually solved by applying some feature selection techniques, such as information gain and χ^2, whereas the latest one is related to the precision of labeling the training data as positive examples and negative examples. In this paper, unless otherwise specified, the term of noise refers to the noise in the training data.

TABLE 6
The Quality of the Reliable Negative Documents

%	Reuters-21578 (10)		Reuters-21578 (90)		Newsgroup-20		WebKB	
	purity	precision	purity	precision	purity	precision	purity	precision
1	0.790	0.890	0.726	0.986	0.741	0.849	0.558	0.858
2	0.811	0.891	0.747	0.987	0.744	0.849	0.559	0.860
3	0.832	0.892	0.797	0.987	0.752	0.850	0.660	0.861
4	0.812	0.893	0.847	0.987	0.751	0.850	0.681	0.862
5	0.853	0.894	0.861	0.987	0.749	0.851	0.742	0.863
6	0.874	0.895	0.841	0.987	0.751	0.851	0.763	0.865
7	0.895	0.896	0.911	0.987	0.752	0.852	0.764	0.866
8	0.896	0.897	0.912	0.987	0.752	0.852	0.765	0.867
9	0.897	0.898	0.917	0.988	0.753	0.853	0.786	0.868
10	0.898	0.899	0.917	0.988	0.753	0.893	0.768	0.869
20	0.906	0.909	0.929	0.989	0.858	0.898	0.779	0.882
30	0.916	0.920	0.920	0.990	0.860	0.963	0.790	0.895
40	0.925	0.930	0.921	0.992	0.838	0.968	0.803	0.909
50	0.936	0.941	0.922	0.993	0.864	0.964	0.786	0.923
60	0.947	0.952	0.924	0.994	0.843	0.959	0.770	0.937
70	0.959	0.964	0.925	0.996	0.834	0.994	0.835	0.952
80	0.971	0.976	0.927	0.997	0.881	0.919	0.822	0.968
90	0.985	0.988	0.928	0.999	0.892	0.925	0.880	0.984
100	0.985	1.000	0.973	1.000	0.900	1.000	0.903	1.000

TABLE 7
The Significance of Step 2 of *PNLH*

%	Reuters-21578 (10)					WebKB				
	SVM	*PNLH*	Case-1	Case-2	Case-3	SVM	*PNLH*	Case-1	Case-2	Case-3
1	0.000	**0.483**	0.426	0.451	0.413	0.000	**0.026**	0.024	0.020	0.004
2	0.000	**0.501**	0.378	0.456	0.371	0.000	**0.035**	0.034	**0.035**	0.004
3	0.000	**0.588**	0.383	0.457	0.376	0.000	0.058	**0.068**	0.062	0.014
4	0.000	**0.625**	0.390	0.535	0.390	0.000	**0.119**	0.071	0.070	0.039
5	0.000	**0.677**	0.532	0.596	0.532	0.000	**0.143**	0.095	0.104	0.053
6	0.000	**0.692**	0.546	0.606	0.567	0.000	**0.348**	0.197	0.191	0.169
7	0.000	**0.716**	0.591	0.650	0.597	0.000	**0.353**	0.214	0.252	0.183
8	0.000	**0.716**	0.614	0.669	0.612	0.000	**0.359**	0.275	0.258	0.235
9	0.000	**0.727**	0.655	0.668	0.654	0.000	**0.404**	0.368	0.402	0.315
10	0.009	**0.729**	0.662	0.669	0.662	0.005	0.374	**0.377**	0.371	0.346
20	0.012	**0.757**	0.726	0.700	0.726	0.029	**0.497**	0.482	0.469	0.413
30	0.080	**0.789**	0.781	0.773	0.781	0.062	**0.539**	0.515	0.528	0.441
40	0.340	0.804	**0.818**	0.764	0.776	0.151	**0.594**	0.580	0.595	0.496
50	0.568	0.824	**0.835**	0.778	0.755	0.260	**0.647**	0.614	0.630	0.526
60	0.737	0.824	**0.857**	0.758	0.792	0.364	**0.675**	0.639	0.639	0.547
70	0.824	0.862	**0.867**	0.756	0.814	0.475	**0.689**	0.647	0.648	0.554
80	0.865	0.874	**0.880**	0.787	0.865	0.544	**0.700**	0.658	0.649	0.629
90	0.865	0.887	**0.891**	0.804	0.878	0.664	0.714	0.675	0.660	**0.715**
100	**0.907**	0.889	0.890	0.816	0.890	0.717	**0.724**	0.679	0.667	0.720

Following the recent trend, we report the micro-Precision and micro-Purity, which is to treat all of the categories as a whole before computing this corresponding values. Table 6 shows the results with different benchmarks. Both precision and purity are very high.

4.4 The Effectiveness of Enlargement (Step 2)

Recall that Step 2 of *PNLH* is to further extract a set of example so as to enlarge the given positive examples and the reliable negative examples. The extraction in Step 2 is based on the distance between the positive examples and the negative examples. To show the significance of the enlargement, we compare *PNLH* with several cases. The first three cases are given below.

- **Case 1**: Extraction only without enlargement.
- **Case 2**: Extraction and enlargement of positive examples only.
- **Case 3**: Extraction and enlargement of negative examples only.

Table 7 shows the results of Reuters-21578 (10) and WebKB. All other benchmarks behave similarly. In Table 7, the column SVM shows the result of SVM without *PNLH* as the baseline for comparison. All the results of *PNLH*, as well as the other three cases, are obtained using SVM after extracting/enlarging examples. *PNLH* significantly outperforms the other three cases for the small cases. This supports our claim and motivation for this paper, such as whether the quality of a classifier can be enhanced by including more positive documents in the training set, especially when $|P|$ is small. For Reuters-21578 (10), from $x = 40$ percent to $x = 100$ percent, *PNLH* degrades slightly in comparison with Case 1 (without enlargement). It is because, as more processing is done, more errors can possibly be introduced. However, the differences are marginal. Also, as shown in Table 7, when $|P|$ is small (say, 1 percent-50 percent), the influence of P' is more important than N' because the Case 2 is superior to Case 3 for small cases. This suggests that enlarging P' is more important than enlarging N' when P is small.

Furthermore, we show the significance of feature selection (Algorithm 3) with an additional case:

TABLE 8
The Necessity of Feature Selection

%	Reuters-21578 (10)		Reuters-21578 (90)		WebKB		Newsgroup-20	
	PNLH	Case-4	*PNLH*	Case-4	*PNLH*	Case-4	*PNLH*	Case-4
1	**0.483**	0.401	**0.375**	0.359	**0.026**	0.000	0.000	0.000
2	**0.501**	0.465	**0.432**	0.174	**0.035**	0.026	**0.026**	0.000
3	**0.588**	0.476	**0.506**	0.384	**0.058**	0.037	**0.040**	0.000
4	**0.625**	0.527	**0.542**	0.439	**0.119**	0.042	**0.069**	0.000
5	**0.677**	0.582	**0.553**	0.469	**0.143**	0.040	**0.094**	0.000
6	**0.692**	0.612	**0.570**	0.488	**0.348**	0.085	**0.185**	0.000
7	**0.716**	0.650	**0.592**	0.517	**0.353**	0.055	**0.237**	0.031
8	**0.716**	0.663	**0.595**	0.529	**0.359**	0.175	**0.274**	0.074
9	**0.727**	0.679	**0.604**	0.557	**0.404**	0.193	**0.314**	0.101
10	**0.729**	0.683	**0.623**	0.587	**0.374**	0.241	**0.343**	0.110
20	**0.757**	0.728	**0.683**	0.614	**0.497**	0.458	**0.423**	0.141
30	**0.789**	0.771	**0.710**	0.687	**0.539**	0.504	**0.444**	0.157
40	**0.804**	0.801	**0.733**	0.718	**0.594**	0.588	**0.437**	0.225
50	**0.824**	0.814	**0.761**	0.756	**0.647**	0.623	**0.450**	0.294
60	**0.824**	**0.824**	**0.796**	0.790	**0.675**	0.649	**0.481**	0.301
70	0.862	**0.864**	**0.792**	0.788	**0.689**	0.658	**0.535**	0.429
80	**0.874**	0.870	**0.828**	0.833	**0.700**	0.668	**0.632**	0.568
90	0.887	**0.889**	0.808	**0.818**	**0.714**	0.683	0.662	**0.685**
100	0.889	**0.892**	0.819	**0.827**	**0.724**	0.689	0.722	**0.746**

- **Case 4**: *PNLH* without feature selection. (L in line-1 of the algorithm *Centroid* (Algorithm 3) is the set of all features in $P \cup N$ instead, without selection and ranking.

We have experimented with three different kinds of the most common feature selection techniques: information gain, χ^2, and mutual information [20], and find that their differences are insignificant or none. The results that we reported in Table 8 are based on information gain. Referring to Table 8, we can see that feature section is an important procedure to ensure higher accuracy. Feature selection is especially critical in the two benchmarks: WebKB and Newsgroup-20. The reason is that the numbers of features in WebKB and Newsgroup-20 are significantly more than Reuters-21578. In WebKB and Newsgroup-20, there are in total around 50,000 and 65,000 different features, whereas there are only around 23,000 features in Reuters-21578. The large number of different features implies that the level of noise presented in the data set can be higher.

4.5 Different Labeling Heuristics Comparison

The labeling heuristics are independent from the classifiers to be used (refer to Fig. 1). In this section, we compare *PNLH* with all the heuristics discussed in Section 3 using SVM as the text classifier. The reasons that we choose SVM are three-fold: 1) SVM does not require feature selection. Other classifiers, such as Rocchio and NB, require feature selection as the preprocessing step. The more preprocessing that is done, the less we can conclude about the significance of a specific labeling approach, as the result of that approach may be improved/deteriorate by the preprocessing. 2) SVM is very sensitive to noise (as shown before). Using SVM as the classifier can give a clear picture about the precision of different labeling heuristics. 3) SVM is the best reported classification algorithm up-to-date. Using SVM can possibly know how "best" a labeling heuristic can be.

In the following, we name each of the combinations as follows: Rocchio+SVM (Roc-SVM), Spy+SVM (Spy-SVM), PEBL+SVM (PEBL-SVM), Weighted-Logistic+SVM (WL+SVM), and PNLH+SVM. Tables 9, 10, 11, and 12 show the results.

TABLE 9
Results of Reuters-21578 (10)

%	PNLH-SVM	Spy-SVM	Roc-SVM	PEBL-SVM	WL-SVM
1	**0.483**	0.000	0.000	0.001	0.000
2	**0.501**	0.000	0.001	0.000	0.000
3	**0.588**	0.001	0.008	0.007	0.000
4	**0.625**	0.014	0.008	0.007	0.000
5	**0.677**	0.163	0.008	0.007	0.090
6	**0.692**	0.149	0.490	0.007	0.108
7	**0.716**	0.193	0.491	0.008	0.135
8	**0.716**	0.297	0.489	0.009	0.146
9	**0.727**	0.345	0.488	0.009	0.144
10	**0.729**	0.290	0.499	0.009	0.165
20	**0.757**	0.402	0.515	0.021	0.325
30	**0.789**	0.683	0.603	0.061	0.444
40	0.804	0.738	**0.808**	0.320	0.535
50	0.824	0.753	**0.844**	0.549	0.590
60	0.824	0.777	**0.860**	0.722	0.647
70	0.862	0.795	**0.864**	0.819	0.713
80	**0.874**	0.829	0.861	0.863	0.758
90	0.887	0.837	0.857	**0.891**	0.792
100	0.889	0.850	0.855	0.891	0.855

TABLE 10
Results of Reuters-21578 (90)

%	PNLH-SVM	Spy-SVM	Roc-SVM	PEBL-SVM	WL-SVM
1	**0.375**	0.000	0.000	0.001	0.000
2	**0.432**	0.000	0.001	0.000	0.000
3	**0.506**	0.000	0.006	0.005	0.000
4	**0.542**	0.001	0.372	0.005	0.000
5	**0.553**	0.014	0.386	0.005	0.090
6	**0.570**	0.263	0.391	0.005	0.108
7	**0.592**	0.215	0.399	0.005	0.135
8	**0.595**	0.222	0.393	0.006	0.146
9	**0.604**	0.297	0.390	0.006	0.144
10	**0.623**	0.345	0.395	0.006	0.165
20	**0.683**	0.290	0.412	0.009	0.325
30	**0.710**	0.402	0.477	0.043	0.444
40	**0.733**	0.683	0.686	0.252	0.535
50	**0.761**	0.738	0.726	0.443	0.590
60	**0.796**	0.777	0.763	0.614	0.647
70	**0.792**	0.780	0.790	0.704	0.713
80	**0.828**	0.813	0.817	0.782	0.758
90	0.808	0.827	0.826	0.827	0.792
100	0.819	0.845	0.843	0.859	0.855

TABLE 11
Results of WebKB

%	PNLH-SVM	Spy-SVM	Roc-SVM	PEBL-SVM	WL-SVM
1	**0.026**	0.000	0.000	0.000	0.000
2	**0.035**	0.000	0.000	0.000	0.000
3	**0.058**	0.000	0.001	0.000	0.000
4	**0.119**	0.000	0.002	0.000	0.000
5	**0.143**	0.002	0.006	0.002	0.000
6	**0.348**	0.007	0.010	0.002	0.000
7	**0.353**	0.006	0.014	0.003	0.000
8	**0.359**	0.007	0.014	0.004	0.000
9	**0.404**	0.009	0.014	0.004	0.000
10	**0.374**	0.011	0.016	0.006	0.000
20	**0.497**	0.053	0.053	0.028	0.000
30	**0.539**	0.078	0.135	0.059	0.168
40	**0.594**	0.154	0.264	0.151	0.237
50	**0.647**	0.235	0.401	0.260	0.389
60	**0.675**	0.517	0.514	0.359	0.420
70	**0.689**	0.581	0.607	0.474	0.561
80	**0.700**	0.697	0.670	0.588	0.594
90	**0.714**	0.698	0.703	0.667	0.658
100	**0.724**	0.710	0.723	0.716	0.704

TABLE 12
Results of Newsgroup-20

%	PNLH-SVM	Spy-SVM	Roc-SVM	PEBL-SVM	WL-SVM
1	0.000	0.000	0.000	0.000	0.000
2	**0.026**	0.000	0.000	0.000	0.000
3	**0.040**	0.001	0.000	0.000	0.000
4	**0.069**	0.000	0.000	0.000	0.000
5	**0.094**	0.000	0.000	0.000	0.000
6	**0.185**	0.001	0.001	0.000	0.000
7	**0.237**	0.000	0.000	0.000	0.047
8	**0.274**	0.002	0.001	0.001	0.054
9	**0.314**	0.001	0.002	0.001	0.064
10	**0.343**	0.001	0.003	0.001	0.171
20	**0.423**	0.042	0.018	0.012	0.243
30	**0.444**	0.070	0.070	0.027	0.304
40	**0.437**	0.160	0.191	0.071	0.362
50	**0.450**	0.252	0.329	0.157	0.423
60	**0.481**	0.328	0.449	0.294	0.473
70	0.535	**0.551**	0.545	0.429	0.522
80	0.632	0.653	**0.638**	0.568	0.579
90	0.662	0.699	**0.702**	0.665	0.644
100	0.722	0.731	**0.766**	0.756	0.696

PEBL-SVM is only acceptable ($F1 > 0.5$) when $|P|$ is large ($|P| = 40$ percent in Reuters-21578 (10), $|P| = 60$ percent in Reuters-21578 (90), $|P| = 80$ percent in WebKB, and $|P| = 80$ percent in Newsgroup-20). WL-SVM outperforms PEBL-SVM, but is inferior to the others. Roc-SVM performs the second best, although Spy-SVM outperforms Roc-SVM in a few cases. PNLH-SVM outperforms all the others significantly, even when $|P|$ is extremely small (≤ 40 percent). It does not vary much regardless of the size of P. In addition, PNLH-SVM performs the best on all cases in Web-KB.

Finally, Table 13 shows some representative results of macro-F1 for the most skewed data set: Reuters21578 (90). Macro-F1 is usually used to evaluate a classifier against skewed data sets. If the documents are evenly distributed among categories, then both micro-F1 and macro-F1 would be similar. Hence, we do not report the macro-F1 for Newsgroup-20 and WebKB because the former one is evenly distributed while the latter one is just slightly skewed. As shown in Table 13, *PNLH* outperforms all other approaches.

5 Conclusion

We present a partition-based heuristic to build a text classifier using only positive examples (P) and unlabeled examples (U). No negative example is given explicitly.

Existing techniques that solve this problem only focus on extracting negative examples (N) from U. They cannot extract a high quality of positive examples (P') from U and, therefore, cannot fully utilize the information in there. Unfortunately, extracting P' from U is difficult. The difficulty is due to the topic diversity in U. In order to solve the diversity problem, we proposed a heuristic which contains two steps: extraction and enlargement.

In Step 1 (extraction), we extract some negative examples N from U based on a concept called core vocabulary and a $G(\cdot)$ function. Core vocabulary is constructed based on a concept called feature strength, which is computed based on a $H(\cdot)$ function with a threshold θ. The $G(\cdot)$ function, together with a threshold ϕ, is used to determine whether a document can be considered as a potential positive example. Note that both θ and ϕ are computed automatically.

In Step 2 (enlargement), we enlarge both the given P and the extracted N by a partition-based heuristic. We partition N into k clusters, so as to assist us extract some positive examples (P') and more negative examples (N') from U' ($U' = U - N$). k is determined dynamically and automatically, such that each category may have different k even in the same domain.

We conducted extensive experiments using three benchmarks: Reuters21578, Newsgroup20, and WebKB. The results indicated that our approach is highly feasible and significantly outperforms the others when $|P|$ is extremely small.

TABLE 13
Some Macro-F1 Results for Reuters21578 (90)

%	SVM	PNLH-SVM	Spy-SVM	Roc-SVM	PEBL-SVM	WL-SVM
1	0.000	**0.012**	0.000	0.000	0.000	0.000
4	0.000	**0.059**	0.000	0.012	0.000	0.000
7	0.000	**0.139**	0.000	0.015	0.000	0.028
10	0.000	**0.211**	0.008	0.016	0.000	0.058
40	0.035	**0.295**	0.110	0.126	0.048	0.213
70	0.224	**0.413**	0.264	0.282	0.258	0.285
100	0.413	**0.458**	0.409	0.417	0.445	0.410

REFERENCES

[1] D. Bennett and A. Demiritz, "Semi-Supervised Support Vector Machines," *Advances in Neural Information Processing Systems,* vol. 11, 1998.

[2] J. Bockhorst and M. Craven, "Exploiting Relations Among Concepts to Acquire Weakly Labeled Training Data," *Proc. 19th Int'l Conf. Machine Learning,* 2002.

[3] P. Bradley and U. Fayyad, "Refining Initial Points for k-Means Clustering," *Proc. 15th Int'l Conf. Machine Learning,* 1998.

[4] D.R. Cutting, D.R. Karger, J.O. Pederson, and J.W. Tukey, "Scatter/Gather a Cluster-Based Approach to Browsing Large Document Collections," *Proc. 15th Int'l Conf. Research and Development in Information Retrieval,* 1992.

[5] G.P.C. Fung, J.X. Yu, H. Lu, and P.S. Yu, "Text Classification without Negative Examples," *Proc. 21st Int'l Conf. Data Eng.,* 2005.

[6] R. Ghani, "Combining Labeled and Unlabeled Data for Multiclass Text Categorization," *Proc. 19th Int'l Conf. Machine Learning,* 2002.

[7] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning,* 1998.

[8] B. Larsen and C. Aone, "Fast and Effective Text Mining Using Linear-Time Document Clustering," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining,* 1999.

[9] W.S. Lee and B. Liu, "Learning with Positive and Unlabeled Examples Using Weighted Logistics Regression," *Proc. 20th Int'l Conf. Machine Learning,* 2003.

[10] X. Li and B. Liu, "Learning to Classify Texts Using Positive and Unlabeled Data," *Proc. 2003 Int'l Joint Conf. Artificial Intelligence,* 2003.

[11] B. Liu, Y. Dai, X. Li, W.S. Lee, and P.S. Yu, "Building Text Classifiers Using Positive and Unlabeled Examples," *Proc. Third Int'l Conf. Data Mining,* 2003.

[12] B. Liu, X. Li, W.S. Lee, and P.S. Yu, "Text Classification by Labeling Words," *Proc. 19th Nat'l Conf. Artificial Intelligence,* 2004.

[13] B. Liu, P.S. Yu, and X. Li, "Partially Supervised Classification of Text Documents," *Proc. 19th Int'l Conf. Machine Learning,* 2002.

[14] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," *Proc. 15th Nat'l Conf. Artificial Intelligence Workshop Learning for Text Categorization,* 1998.

[15] D.C. Montogomery and G.C. Runger, *Applied Statistics and Probability for Engineers,* second ed. John Wiley & Sons, Inc., 1999.

[16] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM," *Machine Learning,* vol. 39, 2000.

[17] J. Rocchio, "Relevance Feedback Information Retrieval," *The Smart Retrieval System—Experiments in Automatic Document Processing,* G. Salton, ed., pp. 313-323. Prentice Hall, 1971.

[18] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management,* vol. 24, no. 5, pp. 513-523, 1988.

[19] H. Schutze, D.A. Hull, and J.O. Pedersen, "A Comparison of Classifiers and Document Representations for the Routing Problem," *Proc. 18th Int'l Conf. Research and Development in Information Retrieval,* 1995.

[20] F. Seabastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys,* vol. 34, no. 1, pp. 1-47, 2002.

[21] Y. Yang, "A Study on Thresholding Strategies for Text Categorization," *Proc. 24th Int'l Conf. Research and Development in Information Retrieval,* 2001.

[22] Y. Yang and X. Liu, "A Re-Examination of Text Categorization Methods," *Proc. 22nd Int'l Conf. Research and Development in Information Retrieval,* 1999.

[23] H. Yu, K.C.-C. Chang, and J. Han, "Heterogeneous Learner for Web Page Classification," *Proc. Second Int'l Conf. Data Mining,* 2003.

[24] H. Yu, J. Han, and K.C.-C. Chang, "PEBL: Positive Example Based Learning for Web Page Classification Using SVM," *Proc. Ninth Int'l Conf. Knowledge Discovery and Data Mining,* 2003.

[25] H. Yu, J. Han, and K.C.-C. Chang, "PEBL: Web Page Classification without Negative Examples," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, 2004.

[26] T. Zhang, "The Value of Unlabeled Data for Classification Problems," *Proc. 17th Int'l Conf. Machine Learning,* 2000.

Gabriel Pui Cheong Fung received the BEng degree (2001) and the MPhil degree (2003), both in systems engineering and engineering management, from the Chinese University of Hong Kong. He is currently a PhD candidate studying at the same university. His research interests include text mining, time series mining, Internet applications and technologies, and database systems. He has published various papers in several related conferences, including VLDB, ICDE, and ICDM. He received the best student paper award at PAKDD'05. He is a member of the ACM.

Jeffrey X. Yu received the BE, ME, and PhD degrees in computer science from the University of Tsukuba, Japan, in 1985, 1987, and 1990, respectively. He was a research fellow (April 1990-March 1991) and a faculty member (April 1991-July 1992) at the Institute of Information Sciences and Electronics, University of Tsukuba, and a lecturer in the Department of Computer Science, Australian National University (July 1992-June 2000). Currently, he is an associate professor in the Department of Systems Engineering and Engineering Management, The Chinese University of Hong Kong. His major research interests include data mining, data streaming mining and processing, data warehouse, online analytical processing, XML query processing, and optimization. He is a member of the IEEE Computer Society

Hongjun Lu received the BSc degree from Tsinghua University, China, and the MSc and PhD degrees from the Department of Computer Science, University of Wisconsin-Madison. Before joining the Hong Kong University of Science and Technology, he worked as an engineer at the Chinese Academy of Space Technology, a principal research scientist in the Computer Science Center of Honeywell Inc., Minnesota (1985-1987), and a professor at the School of Computing at the National University of Singapore (1987-2000). His research interests are in data/knowledge base management systems with an emphasis on query processing and optimization, physical database design, and database performance. His recent research work includes data quality, data warehousing and data mining, and management of XML data. He is also interested in the development of Internet-based database applications and electronic business systems.

Philip S. Yu received the BS degree in electrical engineering from National Taiwan University, the MS and PhD degrees in electrical engineering from Stanford University, and the MBA degree from New York University. He is with the IBM Thomas J. Watson Research Center and is currently the manager of the Software Tools and Techniques group. His research interests include data mining, Internet applications and technologies, database systems, multimedia systems, parallel and distributed processing, and performance modeling. He has published more than 400 papers in refereed journals and conferences and holds or has applied for more than 250 US patents. He is an associate editor of the *ACM Transactions on the Internet Technology*, is a member of the *IEEE Data Engineering* steering committee, and is also on the steering committee of the IEEE Conference on Data Mining. He was the editor-in-chief of *IEEE Transactions on Knowledge and Data Engineering* (2001-2004) and an editor, advisory board member, and guest coeditor of the special issue on mining of databases. He has also served as an associate editor of *Knowledge and Information Systems*. He has been a program chair, cochair, or committee member on many international conferences and has received several IBM honors, including two IBM outstanding innovation awards, an outstanding technical achievement award, two research division awards, and the 84th plateau of invention achievement award. He received an outstanding contributions award from the IEEE International Conference on Data Mining in 2003 and also an IEEE Region 1 Award for "promoting and perpetuating numerous new electrical engineering concepts" in 1999. He is an IBM Master Inventor, a fellow of the ACM, and a fellow of the IEEE.