Query-By-Multiple-Examples using Support Vector Machines

Dell Zhang[1], Wee Sun Lee[2]
[1]SCSIS
Birkbeck, University of London
London WC1E 7HX, UK
dell.z@ieee.org
[2]Department of Computer Science
National University of Singapore
Singapore 117590
leews@comp.nus.edu.sg

Journal of Digital Information Management

ABSTRACT: *We identify and explore an Information Retrieval paradigm called Query-By-Multiple-Examples (QBME) where the information need is described not by a set of terms but by a set of documents. Intuitive ideas for QBME include using the centroid of these documents or the well-known Rocchio algorithm to construct the query vector. We consider this problem from the perspective of text classification, and find that a better query vector can be obtained through learning with Support Vector Machines (SVMs). For online queries, we show how SVMs can be learned from one-class examples in linear time. For offline queries, we show how SVMs can be learned from positive and unlabeled examples together in linear or polynomial time, optimising some meaningful multivariate performance measures. The effectiveness and efficiency of the proposed approaches have been confirmed by our experiments on four real-world datasets.*

Keywords: Information Retrieval, Text Classification, Machine Learning, Support Vector Machine

Received 1 January 2009, Revised 13 March 2009, Accepted 26 March 2009

1. Introduction

Information Retrieval (IR) [20] is basically the science of finding documents that satisfy a given query from a large corpus. In conventional IR systems, a query is just a set of terms that describe the information need. In this paper, we would like to explore an alternative IR paradigm where the information need is described not by a set of terms but by a set of documents [39]. This may sound similar to the concept of Query-By-Example in database management systems [23], but we emphasize the cases where we have multiple examples (documents) instead of just one, and use the term Query- By-Multiple-Examples (QBME) to reflect this emphasis.

Definition 1. ***Query-By-Multiple-Examples (QBME)*** *Given a set of documents $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ as query, retrieve/rank the documents in a corpus $U = \{\mathbf{x}_{l+1}, \ldots, \mathbf{x}_{l+u}\}$ according to their relevance to P.*

The problem of QBME occurs frequently in practice. This is because only relevant documents are usually stored, and it is often desirable to find more relevant documents[1]. For example, a researcher may have saved in her computer some journal articles on a specialized subtopic in bioinformatics *(P),* and she wants to find more materials on that subtopic from the PubMed Central digital library *(U).* For another example, a user searched the Web using a search engine and clicked on some returned links that he was interested in, then the search engine could identify the ads that are most similar to those clicked links *(P)* from its large ad collection *(U),* and show them as "sponsored links" [22].

The intuitive idea for QBME is to use the centroid of these documents (P) or the well-known Rocchio algorithm [24, 20] to construct the query vector. However, given that we have a set of relevant documents which should be more informative than just keyword queries, we may hope to obtain a better query vector by considering this problem from the perspective of *text classification.* Actually we can say that QBME crosses the traditional boundary of retrieval and classification. One of the most prominent *machine learning* techniques for classification is Support Vector Machine (SVM) [27] which has solid theoretical basis and broad practical success. SVM in its simplest form, linear SVM, consistently provides state-of-the-art performance for various text classification tasks [4, 8, 13, 29, 33].

We distinguish between two types of queries: *online* queries where response is required immediately and *offline* queries where the user is willing to wait in order to get better results. For online queries, it is desirable to construct the query quickly and to have a *sparse* representation of query so that the text index (such as inverted file) can be utilized to retrieve relevant documents quickly [20]. For offline queries, computational-intensive methods that process a significant portion of the corpus may be acceptable.

For online queries, we restrict ourselves to learning only from the documents in the query set *(P)* because of efficiency consideration. We show that one-class linear SVM can be trained in time *linear* in $|P|$ and yield a sparse query vector, allowing efficient processing of online queries when the query set size is moderately large. For offline processing, in addition to the query set *(P)*, it may be feasible to use the entire corpus *(U)* for training to improve performance. We examine some *meaningful multivariate performance measures for learning with positive and unlabeled examples* and show that SVMs can be trained in time linear or polynomial in $|P \cup U|$ for these performance measures. Our experiments on four real-world datasets have confirmed the effectiveness and efficiency of the proposed approaches.

1 QBME is not necessarily restricted to text data—it can also be applied to multimedia IR as well.

The rest of this paper is organized as follows. We first propose our approaches to QBME (in Section II), then present experimental evaluation (in Section III), later discuss related work (in Section IV), and finally make conclusions (in Section V).

2. Approaches

We take the classic *Vector Space Model* [24, 20] for this problem of QBME: every document in *P* or *U* is represented as an m-dimensional vector where each dimension corresponds to a term. Moreover, every document vector is *normalized* to unit length. Thus, if the query *P* can also be described as a vector w as in the traditional IR paradigm, effective retrieval can be achieved by ranking the documents $\mathbf{x} \in U$ according to their cosine similarities with w, i.e., $\cos(\theta)$ where θ is the angle between w and x. Since

$$\cos(\theta) = \frac{\mathbf{w}}{||\mathbf{w}||} \cdot \frac{\mathbf{x}}{||\mathbf{x}||} = \frac{1}{||\mathbf{w}||}\mathbf{w}^T\mathbf{x} \propto \mathbf{w}^T\mathbf{x},$$

this retrieval strategy is actually ranking the documents in *U* by a linear function

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x},$$

i.e., document $\mathbf{x}_i$ is ranked higher than document $\mathbf{x}_j$ if and only if $\mathbf{w}^T\mathbf{x}_i > \mathbf{w}^T\mathbf{x}_j$. So the problem of QBME is in fact to find the best query vector w.

A. Learning from P Only

1) The Centroid Algorithm

Maybe the simplest method for this problem is to use the centroid of *P*, i.e., the mean of all vectors in *P,* as the query vector:

$$\mathbf{w} = \frac{1}{l}\sum_{i=1}^{l}\mathbf{x}_i.$$

Obviously the algorithm for computing the centroid of *P* only needs one sequential scan of *P*, so its time complexity is $O(l)$.

2) One-Class SVM

If all relevant documents are within a *compact* region in the vector space, we can define the query vector based on that region. Such a region of relevant documents can be learned using P as the set of training examples. It is reasonable to expect that the more accurate the region is, the better the corresponding query vector would be.

OP 1. SVM1c

$$\begin{aligned} \min_{\mathbf{w},\xi_i \geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{l}\sum_{i=1}^{l}\xi_i \\ s.t. \quad & \forall_{i=1}^{l} : \mathbf{w}^T\mathbf{x}_i \geq 1 - \xi_i \end{aligned}$$

In its simplest form, SVM_{1c} attempts to find the query vector w with the smallest norm $||\mathbf{w}||$ that will give a score $\mathbf{w}^T\mathbf{x}$ of at least one for all documents in the query set *P*. By introducing the slack variables ξ_i, some documents in the query set *P* are allowed to have scores less than one in order to find a simpler (smaller-norm) query vector w.

It is known that such a w can be expressed as $\mathbf{w} = \sum_{i=1}^{l}\alpha_i\mathbf{x}_i$ where $\alpha_i \geq 0$ [27]. Following the KKT conditions, non-zero coefficients α_i can only occur if the corresponding vector $\mathbf{x}_i$ lies on the boundary of *P*. Those vectors with non-zero coefficients α_i are called *support vectors*. Hence we see that the query vector w obtained by learning with SVMs from *P* is essentially in the direction of a *weighted* mean of support vectors in *P*. Furthermore, such a query vector w is more *sparse* than the centroid, i.e., it has fewer non-zero entries. To retrieve/rank the documents in *U*, it is necessary to know the dot-product between w and each $\mathbf{x}_i \in U$. A common way to accelerate this process is to utilize the *inverted index* of U: we walk through the postings in the inverted index for the terms with non-zero weights in w, and then accumulating the total score for each document by "postings merge" [20]. Therefore the retrieval time for an online query depends not only on the training time but also on the sparsity of the query vector.

Note that our SVM_{1c} formulation here is different with the original one-class SVM [26, 27] which relies on a parameter ν (as in ν-SVM [28, 27]) but not *C*. The parameter ν has more intuitive sense than C: it directly controls the amount of margin errors and the number of support vectors. Nevertheless, the classification function of the original oneclass SVM optimised for a given ν would be same as that of SVM_{1c} with a certain value of *C*.

Current training methods for ν -SVMs (including the original one-class SVM) take at least quadratic time. With our SVM_{1c} formulation, we can achieve linear-time training by transforming it to a special case of SVM^{struct} — the structural SVM formulation which was first proposed for training SVMs to predict structural outputs [10].

OP 2. SVM_{1c}^{struct}

$$\begin{aligned} \min_{\mathbf{w},\xi \geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\ s.t. \quad & \forall \bar{\eta} \in \{0,1\}^l : \\ & \frac{1}{l}\mathbf{w}^T\sum_{i=1}^{l}\eta_i\mathbf{x}_i \geq \frac{1}{l}\sum_{i=1}^{l}\eta_i - \xi \end{aligned}$$

Proposition 1. SVM_{1c} *is equivalent to* SVM_{1c}^{struct}

Proof. Adapting the proof from [10], we will show that any solution $\mathbf{w}^*$ of SVM_{1c} is also a solution of SVM_{1c}^{struct} and vice versa.

For a given w, the ξ_i in SVM_{1c} can be optimised individually by $\xi_i = \max\{0, 1 - \mathbf{w}^T\mathbf{x}_i\}$.

For the same w, the optimal ξ in SVM_{1c}^{struct} is

$$\xi = \max_{\bar{\eta}\in\{0,1\}^l}\left\{\frac{1}{l}\sum_{i=1}^{l}\eta_i - \frac{1}{l}\sum_{i=1}^{l}\eta_i\mathbf{w}^T\mathbf{x}_i\right\}.$$

Since this function is linear in η_i, each η_i can be optimised independently. Therefore the optimal ξ and $\sum_{i=1}^{l}\xi_i$ are related as follows.

$$\begin{aligned} \min \xi &= \sum_{i=1}^{l}\max_{\eta_i\in\{0,1\}}\left\{\frac{1}{l}\eta_i - \frac{1}{l}\eta_i\mathbf{w}^T\mathbf{x}_i\right\} \\ &= \sum_{i=1}^{l}\max\left\{0, \frac{1}{l} - \frac{1}{l}\mathbf{w}^T\mathbf{x}_i\right\} \\ &= \min\frac{1}{l}\sum_{i=1}^{l}\xi_i \end{aligned}$$

It means that the objective values of SVM^{1c} and SVM_{1c}^{struct} are equal for any w given the optimal ξ and ξ_i, consequently so are their optima.

Proposition 2. SVM_{1c}^{struct} *(*SVM_{1c}^{struct}*) can be trained in linear time w.r.t. the size of P, i.e., l.*

Proof. It has been shown that SVM_{1c}^{struct} can be correctly trained by the *cutting-plane algorithm* [11] in $O(sl)$ time where s is the average number of non-zero features and l is the number of training examples [9]. SVM_{1c}^{struct} is just a special case of SVM^{struct} that all training examples happen to be positive, therefore it can be trained using the same algorithm as well.

B. Learning from *P* and *U*

1) The Rocchio Algorithm

Another simple method for QBME is to use the Rocchio algorithm [20, 24]. As in (pseudo) relevance feedback [20, 24], given a set of documents *P* that is indicated to be relevant and the corpus *U*, the simplified Rocchio algorithm (aka the Prototype algorithm [14]) constructs the following query vector from both *P* and *U* to rank all the documents:

$$\mathbf{w} = \frac{1}{l}\sum_{i=1}^{l}\mathbf{x}_i - \frac{1}{u}\sum_{j=1}^{u}\mathbf{x}_j.$$

The Rocchio algorithm only needs one sequential scan of *P* and *U*, therefore its time complexity is $O(l+u)$ If the mean vector of *U* has been pre-computed, the Rocchio algorithm for a query can be done in $O(l)$, but such preprocessing is not always possible.

2) SVMs for PU Learning

Suppose that we know not only a set of relevant documents *P*, but also a set of irrelevant documents *N*, we may be able to exploit both sets to get a region of relevant documents which is more accurate than that obtained from *P* only. Such a region of relevant documents can be learned by training a classifier (such as SVM) taking documents in *P* as *positive* training examples and documents in *N* as *negative* training examples. The obstacle of this method is that *N* is not available in the context of QBME. What we have in addition to *P* is only the unlabeled corpus *U* where relevant and irrelevant documents are mixed. This problem of training a classifier using positive and unlabeled examples is called *PU Learning* [16].

If we take the relevant documents in *U* as noise, then we can consider *U* as a very noisy set of negative training examples. There are totally $n = l + u$ examples. Denote the *observed* label of an example x by $y \in \{-1, +1\}$, i.e., $\forall \mathbf{x}_i \in P : y_i = 1$ and $\forall \mathbf{x}_i \in U : y_i = -1$. Denote the *actual* label of an example x by $z \in \{-1, +1\}$ that indicates its true relevancy. We know that $\forall \mathbf{x}_i \in P : z_i = y_i = 1$, but we have no idea about the *hidden* value of z_i for any $\mathbf{x}_i \in U$.

It is reasonable to assume that the documents in *P* are randomly sampled from the class of relevant documents with a constant but unknown probability $\Pr[y = 1|z = 1] = \mu$. In other words, an actual relevant document has probability μ. to be observed in *P* and probability $1 - \mu$ to be left in *U*; while all actual irrelevant documents are put in *U*. We have the following relationships:

$$\begin{aligned}\Pr[y = +1] &= \Pr[z = +1]\mu;\\ \Pr[y = -1] &= \Pr[z = -1] + \Pr[z = +1](1 - \mu).\end{aligned}$$

When the document collection *U* contains no relevant documents, i.e., $\forall \mathbf{x}_i \in U : z_i = y_i = -1$, we can use the following standard SVM formulation to get a good classifier and consequently a good query vector.

OP 3. SVM_{2C}

$$\begin{aligned}\min_{\mathbf{w},\xi_i \geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i\\ s.t. \quad & \forall_{i=1}^{n} : y_i\mathbf{w}^T\mathbf{x}_i \geq 1 - \xi_i\end{aligned}$$

SVM_{2C} attempts to find a hyperplane[2] $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that can separate relevant document vectors (positive examples) and irrelevant document vectors (negative examples) with a *large margin* $\frac{2}{||\mathbf{w}||}$ as well as small empirical *hinge* loss $\sum_{i=1}^{n}\xi_i$. Then the *normal vector* of this plane, w, can be used as the query vector.

It is known that such a w can be expressed as $\mathbf{w} = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i$ where $\alpha_i \geq 0$ [27]. Following the KKT conditions, non-zero coefficients α_i can only occur if the corresponding vector $\mathbf{x}_i$ lies on the interface between *P* and *U*. Those vectors with non-zero coefficients α_i are called *support vectors*. Hence we see that the query vector w obtained by learning with SVMs from *P* and *U* is essentially in the direction of a *weighted* mean of *support vectors* in *P* and *U*. However, in real-world IR applications, the corpus *U* usually contains a number of relevant documents (positive examples). In these situations, applying the above SVM_{2C} straightforwardly to the problem of QBME would not work. Let's call the classification performance calculated over observed (noisy) labels y_i *observed* performance, and the classification performance calculated over actual labels z_i *actual* performance. SVM_{2C} minimises the observed error rate, but low observed error rate does not necessarily lead to low actual error rate [1]. For example, when there are 100 documents in *U* relevant to the given query *P* that consists of 10 documents, the actual optimal classifier h_1 would generate 100 observed errors, in contrast, another classifier h_2 that classifies all examples to be negative would generate 10 observed errors, consequently h_2 is favoured by SVM_{2C} over the actual optimal classifier h_1.

Our key insight is that we are able to train SVMs in the PU Learning setting effectively, if we substitute some other multivariate performance measures for error rate. The SVM formulation needs to be extended first as follows.

OP 4. SVM^{perf}

$$\begin{aligned}\min_{\mathbf{w},\xi \geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi\\ s.t. \quad & \forall \bar{y}' \in \{+1, -1\}^n \setminus \bar{y} :\\ & \frac{1}{2n}\mathbf{w}^T\sum_{i=1}^{n}(y_i - y_i')\mathbf{x}_i \geq \frac{1}{2n}\Delta(\bar{y}', \bar{y}) - \xi\end{aligned}$$

Joachims has proposed the above SVM^{perf} formulation that directly minimises the loss function Δ corresponding to a multivariate performance measure [9].

Note that our SVM^{perf} formulation here has a slight difference with its original version [9]: a constant factor $\frac{1}{2n}$ is introduced to the constraints in order to better capture how *C* scales with training set size [27, 10]. As we will see later this also makes the connection between SVMperf and SVM^{struct} [10] more obvious.

Now we propose a new principled approach to PU Learning based on SVMs. For the purpose of this paper, we focus on the multivariate performance measures that can be computed from the *contingency table*.

	$y = +1$ $(\mathbf{x} \in P)$	$y = -1$ $(\mathbf{x} \in U)$
$h(\mathbf{x}) = +1$	a	b
$h(\mathbf{x}) = -1$	c	d

Given *P* and *U*, the values in such a contingency table must satisfy the constraints $a, b, c, d \geq 0$, $a + c = l$ and $b + d = u$. Although there are $n! = (l + u)!$ different rankings, there are only $(l + 1)(u + 1)$ different contingency tables that are legitimate. This makes efficient optimisation feasible.

Balanced Accuracy

The *balanced accuracy* of a classifier is the arithmetic average of sensitivity and *specificity* [30]. It is also known as the Area Under the ROC Curve (AUC) for just one run [30].

2 For simplicity of discussion, all the subsequent SVM formulations in this paper are in their unbiased form. This is not a substantial restriction, because a bias can be easily introduced by adding an additional artificial feature of constant value 1 to each example x [9, 10].

The *actual* balanced accuracy is

$$B = \frac{\Pr[h(\mathbf{x}) = 1|z = 1] + \Pr[h(\mathbf{x}) = -1|z = -1]}{2}.$$

The *observed* balanced accuracy is

$$\widehat{B} = \frac{\Pr[h(\mathbf{x}) = 1|y = 1] + \Pr[h(\mathbf{x}) = -1|y = -1]}{2}.$$

Proposition 3. $\widehat{B} - \frac{1}{2} \propto B - \frac{1}{2}$.

Proof. It can be shown with some simple calculation that [1]

$$\begin{aligned} & (2\widehat{B} - 1)\Pr[y = 1]\Pr[y = -1] \\ = \; & (2B - 1)\Pr[z = 1]\Pr[z = -1]\mu \end{aligned}$$

Therefore, we have

$$\begin{aligned} \widehat{B} - \frac{1}{2} &= (B - \frac{1}{2})\mu\frac{\Pr[z = 1]\Pr[z = -1]}{\Pr[y = 1]\Pr[y = -1]} \\ &\propto B - \frac{1}{2} \end{aligned}$$

This proposition implies that we can optimise the actual balanced accuracy B by optimising the observed balanced accuracy b $\widehat{B}$.

Since $0 \leq \widehat{B} \leq 1$, we define the corresponding multivariate loss function as

$$\Delta_{ba}(\bar{h}(\bar{\mathbf{x}}), \bar{y}) = 1 - \widehat{B}.$$

Let SVM_{ba}^{perf} denote the SVM^{perf} with the loss function Δ_{ba}.

We show that SVM_{ba}^{perf} is equivalent to $\mathrm{SVM}_{we}^{struct}$ — the structural SVM formulation that has been extended by us. The difference between $\mathrm{SVM}_{we}^{struct}$ and its the original version SVM^{perf} [10] is that $\mathrm{SVM}_{we}^{struct}$ assigns different weights λ_i to errors on different training examples $\mathbf{x}_i$.

OP 5. $\mathrm{SVM}_{we}^{struct}$

$$\begin{aligned} \min_{\mathbf{w},\xi\geq 0} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\ s.t. \quad & \forall \bar{\eta} \in \{0,1\}^n \setminus \bar{0} : \\ & \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} \eta_i y_i \mathbf{x}_i \geq \frac{1}{n}\sum_{i=1}^{n} \eta_i \lambda_i - \xi \end{aligned}$$

Proposition 4. SVM_{ba}^{perf} *can be reduced to a specific* $\mathrm{SVM}_{we}^{struct}$ *with the following error weight setting:*

$$\lambda_i = \begin{cases} 1/(4l) & \text{if } y_i > 0 \\ 1/(4u) & \text{if } y_i < 0 \end{cases}$$

Proof. SVM_{ba}^{perf} and $\mathrm{SVM}_{we}^{struct}$ have the same objective function to optimise, so we only need to show that they have the equivalent set of constraints.

For each $\bar{y}' \in \{+1, -1\}^n \setminus \bar{y}$, there is a unique corresponding $\bar{\eta} \in \{0,1\}^n \setminus \bar{0}$ through the following one-to-one map:

$$\eta_i = \begin{cases} 0 & \text{if } y_i{}' = y_i \\ 1 & \text{if } y_i{}' \neq y_i \end{cases}$$

So $(y_i - y_i{}')/2 = \eta_i y_i$, and the left-hand-expression of each inequality constraint in SVM_{ba}^{perf} is same as that in $\mathrm{SVM}_{we}^{struct}$. Now let's look at the right-hand-expression of each inequality constraint. Noticing $c = \sum_{y_i>0} \eta_i$ and $d = \sum_{y_i<0} \eta_i$, we can re-write $\sum_{i=1}^{n} \eta_i \lambda_i$ as

$$\begin{aligned} \sum_{y_i>0} \eta_i\lambda_i + \sum_{y_i<0} \eta_i\lambda_i &= \frac{1}{4l}\sum_{y_i>0} \eta_i + \frac{1}{4u}\sum_{y_i<0} \eta_i \\ &= \frac{c}{4(a+c)} + \frac{b}{4(b+d)} \\ &= \frac{1}{2}\Delta_{ba}(\bar{h}(\bar{\mathbf{x}}), \bar{y}) \end{aligned}$$

we see that the right-hand-expression of each inequality constraint in SVM_{ba}^{perf} also turns out to be same as that in $\mathrm{SVM}_{we}^{struct}$. Hence both optimisation problems would lead to the same solution $\mathbf{w}^*$.

Proposition 5. SVM_{ba}^{perf} $(\mathrm{SVM}_{we}^{struct})$ *can be trained in linear time w.r.t. the size of* $P \cup U$, *i.e., l + u.*

Proof. It has been shown that SVM^{struct} can be correctly trained by the *cutting-plane algorithm* [11] in $O(sn)$ time where *s* is the average number of non-zero features and *n* is the number of training examples [9]. The same algorithm can be adapted for the training of $(\mathrm{SVM}_{we}^{struct})$ with little modification. In our case, there are $n = |P \cup U| = l + u$ training examples.

Precision-Recall Product

Rather than accuracy, an IR system is more often evaluated in terms of *precision* and *recall* [20].

The *actual* precision and recall are

$$p = \Pr[z = 1|h(\mathbf{x}) = 1] \text{ and } r = \Pr[h(\mathbf{x}) = 1|z = 1]$$

respectively.

The *observed* precision and recall are

$$\hat{p} = \Pr[y = 1|h(\mathbf{x}) = 1] \text{ and } \hat{r} = \Pr[h(\mathbf{x}) = 1|y = 1]$$

respectively.

Generally speaking, we want both precision and recall to be high in a retrieval situation. The F_1 score, which addresses precision and recall equally, is probably the most popular multivariate performance measure in IR [20]. It is defined as the harmonic average of precision and recall,

$$F_1 = \frac{2}{1/p + 1/r} = \frac{2pr}{p + r}.$$

Unfortunately in PU Learning, high observed F_1 does not guarantee high actual F_1.

We turn to optimise an alternative multivariate performance measure — *pr*, the product of precision and recall. Similar to the F_1 score, *pr* is high when both precision and recall are high, but low when either of them is low. In fact *pr* correlates with F_1 closely. It is easy to see that *pr* is a lower-bound of F_1 and $\sqrt{pr}$ is an upper-bound of F_1.

Proposition 6. $pr \leq F_1 \leq \sqrt{pr}$.

Proof. $F_1 \geq pr$ because $0 \leq p + r \leq 2$. $F_1 \leq \sqrt{pr}$ because the harmonic average can never be greater than the geometric average.

The relationship between the observed precision/recall and the actual precision/recall is given by the following two propositions.

Proposition 7. $\hat{r} = r$.

Proof. This comes directly from the assumption that the examples in *P* is *randomly* sampled from the class of actual positive examples.

$$\begin{aligned} \hat{r} &= \Pr[h(\mathbf{x}) = 1|y = 1] \\ &= \frac{\Pr[h(\mathbf{x}) = 1, y = 1]}{\Pr[y = 1]} \\ &= \frac{\Pr[h(\mathbf{x}) = 1, z = 1]\mu}{\Pr[z = 1]\mu} \\ &= \Pr[h(\mathbf{x}) = 1|z = 1] \\ &= r. \end{aligned}$$

Proposition 8. $\hat{p} \propto p$.

Proof.

$$\begin{aligned}\hat{p} &= \Pr[y=1|h(\mathbf{x})=1] \\ &= \frac{\Pr[h(\mathbf{x})=1|y=1]\Pr[y=1]}{\Pr[h(\mathbf{x})=1]} \\ &= \frac{\hat{r}\Pr[z=1]\mu}{\Pr[h(\mathbf{x})=1]} = \frac{r\Pr[z=1]\mu}{\Pr[h(\mathbf{x})=1]} \\ &= \frac{\Pr[z=1,h(\mathbf{x})=1]}{\Pr[h(\mathbf{x})=1]}\mu \\ &= \Pr[z=1|h(\mathbf{x})=1]\mu \\ &= p\mu \propto p.\end{aligned}$$

Combining the above two propositions, we have the following proposition.

Proposition 9. $\widehat{pr} \propto pr$.

Proof. It is simply because $\hat{p} \propto p$ and $\hat{r} = r$.

This proposition implies that we can optimise the actual precision-recall product pr by optimising the observed precision-recall product $\widehat{pr}$.

Since $0 \leq \widehat{pr} \leq 1$, we define the corresponding multivariate loss function as

$$\Delta_{pr}(\bar{h}(\bar{\mathbf{x}}), \bar{y}) = 1 - \widehat{pr} = 1 - \frac{a^2}{(a+b)(a+c)}.$$

Let SVM^{perf}_{pr} denote the SVM^{perf} with the loss function Δ_{pr}.

Proposition 10. SVM^{perf}_{pr} *can be trained in polynomial time w.r.t. the size of* $P \cup U$*, i.e.,* $l+u$.

Proof. It has been shown that if the loss function Δ can be computed from the contingency table, SVM^{perf} can be correctly trained by a *sparse-approximation algorithm* [32] in $O(n^2 t)$ time where *n* is the number of training examples and *t* is the number of different contingency tables [9]. $\Delta_{pr}(\bar{h}(\bar{\mathbf{x}}), \bar{y})$ can be computed from the contingency table. In our case, there are $n = |P \cup U| = l+u$ training examples, and as we have said, there can only be $(l+1)(u+1) \in O(n^2)$ different legitimate contingency tables.

Precision/Recall at k

In some IR applications, we are interested in the precision or recall on the top few (say k) returned documents [9, 20], which we denote as $p@k$ or $r@k$. For a Web search engine, users care about $p@k$ because they want to find some relevant pages by scanning only the first few (e.g., 10) links in the search result list. For an archival retrieval system, users care about $r@k$ because they hope to find more fraction of all relevant documents by reading only a small number of returned documents. It can be easily seen that $p@k$ and $r@k$ are actually proportional to each other, therefore the optimisation for one of them would also lead to the optimisation for the other simultaneously.

Proposition 8 implies that we can optimise the actual $p@k$ by optimising the observed $\hat{p}@k$.

Since $0 \leq \hat{p}@k \leq 1$, we define the corresponding multivariate loss function as

$$\Delta_{p@k}(\bar{h}(\bar{\mathbf{x}}), \bar{y}) = 1 - \hat{p}@k = 1 - \frac{a}{a+b} = \frac{b}{a+b}$$

for legitimate contingency tables satisfying $a+b=k$.

Proposition 7 implies that we can optimise the actual $r@k$ by optimising the observed $\hat{r}@k$.

Since $0 \leq \hat{r}@k \leq 1$, we define the corresponding multivariate loss function as

$$\Delta_{r@k}(\bar{h}(\bar{\mathbf{x}}), \bar{y}) = 1 - \hat{r}@k = 1 - \frac{a}{a+c} = \frac{c}{a+c}$$

for legitimate contingency tables satisfying $a+b=k$.

Similar to Proposition 10, it is easy to see that $\mathrm{SVM}^{perf}_{p@k}$ and $\mathrm{SVM}^{perf}_{r@k}$ can also be trained in polynomial time w.r.t. the size of $P \cup U$, i.e., $l+u$.

Precision-Recall Break Even Point (PRBEP) [20] is another multivariate performance measure that is often used to evaluate IR systems. For a given query, the precision of the IR system usually decreases along with the increasing of its recall. At a certain point of the precision-recall curve the precision and the recall would be equal to each other, and their value is called the PRBEP. As is obvious from the definition of precision and recall, this equality is achieved when the number of returned documents is just the number of relevant documents. Therefore the observed PRBEP is the observed precision or recall at $k=l$, and the actual PRBEP is the actual precision or recall at $k=l/\mu$. Unfortunately, optimising for the actual PRBEP does not appear possible as μ is unknown.

3. Experiments

A. *Code*

We have implemented all the proposed SVM learning algorithms on the basis of Joachim's SVM^{perf}[3] package. The source code is available at the 1st author's homepage[4].

B. Data

We conduct our experiments on the following four real-world datasets which are pre-processed and publicly available[5,6].

- The **news20** dataset contains approximately 20,000 articles that were collected from 20 different newsgroups.
- The **siam-competition2007** dataset contains 28,596 aviation safety reports that were used in the SIAM Text Mining Competition 2007.
- The **mediamill-exp1** dataset contains 43,907 camerashots from 85 hours of international news broadcast video data that were used in the MediaMill Challenge Problem for generic video indexing.
- The **reuters-21578** dataset contains more than 10,000 labelled news stories [33].

Method	Mean PRBEP	Mean Average Precision
Centroid	0.4299	0.4011
SVM_{1c}	0.3874	0.3436
Rocchio	0.6628	0.6867
SVM^{perf}_{ba}	0.7969	0.8200
SVM^{perf}_{pr}	0.7976	0.8236

Table 1. The effectiveness of QBME methods on the 20news dataset.

Method	Mean PRBEP	Mean Average Precision
Centroid	0.2453	0.2115
SVM_{1c}	0.2618	0.2239
Rocchio	0.4754	0.4627
SVM^{perf}_{ba}	0.5685	0.5635
SVM^{perf}_{pr}	0.5613	0.5565

Table 2. The effectiveness of QBME methods on the siamcompetition2007 dataset.

3 http://svmlight.joachims.org/svm perf.html
4 http://www.dcs.bbk.ac.uk/206dell/
5 http://www.csie.ntu.edu.tw/206cjlin/libsvmtools/datasets/
6 http://www.cs.cmu.edu/206hustlf/r21578 vec download.html

C. Tasks

For each dataset, we construct the retrieval tasks based on the recommended training/testing split. Given a retrieval topic, the query *P* consists of the relevant documents before the split point, while the corpus *U* consists of all (relevant and irrelevant) documents after the split point. Therefore *U* always has a fixed number of documents, while the size of *P* varies with different retrieval topics. On the mediamill-exp1 dataset the top 5 semantic concepts are used as retrieval topics. On the reuters-21578 dataset the 65 semantic concepts with more than five documents are used as retrieval topics. Note that in contrast to the text classification experiments such as in [33, 36], we discard all irrelevant documents before the split point in each of our experiments. As we do not make use of any negative labelled examples, the experimental results reported here can not be directly compared to those in text classification setting.

D. Parameters

We simply set the SVM parameter *C* = 100 throughout all our experiments. Tuning the SVM parameter for specific SVM learning algorithms or specific datasets would almost certainly lead to better retrieval performance, but extensive parameter tuning is computationally too expensive to be practical in the context of retrieval.

E. Results

We evaluate[7] the retrieval effectiveness of each QBME method with Mean PRBEP [20] and Mean Average Precision (MAP[8]) [20], as shown in Table 1, 2, 3 and 4.

The PRBEP scores of each QBME method on the news20 and siam-competition2007 datasets are also shown in Figure 1 and 2 respectively, where the the queries are ordered by their hardness (baseline performance).

Method	Mean PRBEP	Mean Average Precision
Centroid	0.4627	0.4747
SVM_{1c}	0.4705	0.4835
Rocchio	0.5364	0.5555
SVM_{ba}^{perf}	0.6472	0.6735
SVM_{pr}^{perf}	0.6493	0.6727

Table 3. The effectiveness of QBME methods on the mediamill-exp1 dataset.

Method	Mean PRBEP	Mean Average Precision
Centroid	0.6278	0.6833
SVM_{1c}	0.6531	0.6974
Rocchio	0.6474	0.7138
SVM_{ba}^{perf}	0.6621	0.7194
SVM_{pr}^{perf}	0.6711	0.7289

Table 4. The effectiveness of QBME methods on the reuters- 21578 dataset.

It turns out that as we have anticipated learning from both *P* and *U* leads to much higher performances than learning from only *P*. For the methods using only *P*, SVM_{1c} and the Centroid algorithm have similar performances. For the methods using both *P* and *U*, SVM_{ba}^{perf} and SVM_{pr}^{perf} have similar performances, and they both work substantially better than the Rocchio algorithm. According to one-sided t- test [21, 33], the effectiveness superiority of SVM_{ba}^{perf} or SVM_{pr}^{perf} over the Rocchio algorithm is at the significance level 99.5% (P_value < 0:005).

We also find that there is more variance in retrieval performance for an individual QBME method across different queries than for an individual query across different QBME methods. This is consistent with previous findings on traditional keyword query processing [20]. It seems that SVMs make most difference with the baseline algorithms on moderately hard queries.

The retrieval efficiency mainly depends on two factors: (1) the training time to construct query vectors and (2) the sparsity of constructed query vectors. The query vector spar-sity can be measured by the average number of none-zeroentries — the fewer none-zero-entries in a query vector, the shorter the retrieval time would be because the system can perform document similarity computations faster using some indexes [20]. We run our experiments on a PC with Pentium 4 (3GHz) processor and 2GB memory, and report the average CPU seconds of training as well as query vector sparsity for each QBME method in Table 5, 6, 7 and 8.

Method	Training Seconds	Query Vector Sparsity
Centroid	—	11533
SVM_{1c}	0.0250	5179
Rocchio	—	41023
SVM_{ba}^{perf}	0.8905	22919
SVM_{pr}^{perf}	7.1325	22943

Table 5: The efficiency of QBME methods on the news20 dataset.

Method	Training Seconds	Query Vector Sparsity
Centroid	—	9627
SVM_{1c}	0.0327	3597
Rocchio	—	19146
SVM_{ba}^{perf}	1.0409	12258
SVM_{pr}^{perf}	43.0641	12348

Table 6. The efficiency of QBME methods on the siamcompetition2007 dataset.

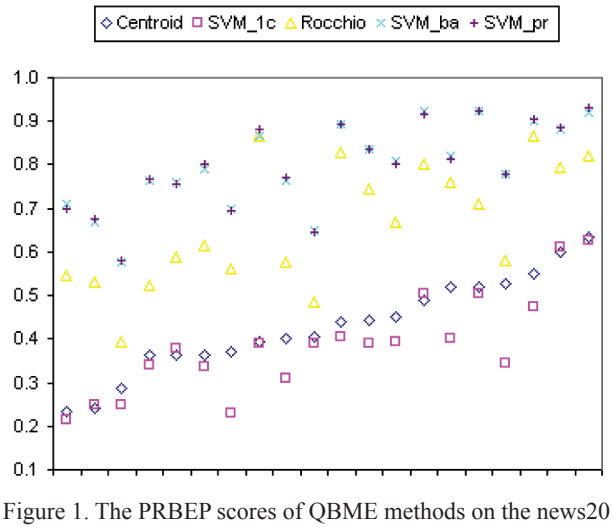

Figure 1. The PRBEP scores of QBME methods on the news20 dataset.

It turns out that all the proposed SVM algorithms for QBME can be trained with a fast speed. Not surprisingly the training time of SVM_{1c} is much shorter than SVM_{ba}^{perf} or SVM_{pr}^{perf}. The running time of either SVM_{ba}^{perf} or SVM_{pr}^{perf} will increase with the size of the corpus $|U|$, which makes them

7 Since the Centroid algorithm for QBME only gives a ranking of documents, we are not able to use F1, Balanced Accuracy or Precision-Recall Product etc. for performance comparison.

8 MAP is the most popular retrieval performance measure among the TREC community. For one query, the average precision is defined to be the average of precisions obtained for the top set of documents existing after each relevant document is retrieved. The MAP value for a set of queries is the arithmetic mean of average precisions for individual queries.

more appropriate for offline query processing when $|U|$ is large. Alternatively, sampling *U* may provide a smooth trade-off between retrieval performance and running time, and allow some use of these methods in online query processing. Moreover, we also see that SVM_{ba}^{perf} runs an order of magnitude faster than SVM_{pr}^{perf} (except on reuters-21578). The linear time complexity of SVM_{ba}^{perf} makes it more scalable than SVM_{pr}^{perf}.

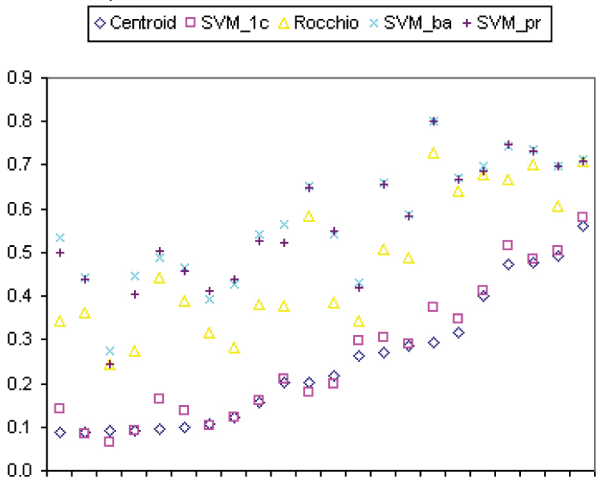

Figure 2. The PRBEP scores of QBME methods on the siam-competition2007 dataset.

Method	Training Seconds	Query Vector Sparsity
Centroid	—	120
SVM_{1c}	0.1560	120
Rocchio	—	120
SVM_{ba}^{perf}	3.6140	120
SVM_{pr}^{perf}	731.3120	120

Table 7. The efficiency of QBME methods on the mediamillexp1 dataset.

Method	Training Seconds	Query Vector Sparsity
Centroid	—	1306
SVM_{1c}	0.0169	1048
Rocchio	—	7241
SVM_{ba}^{perf}	1.8042	4786
SVM_{pr}^{perf}	1.9352	4391

Table 8. The efficiency of QBME methods on the reuters- 21578 dataset.

Furthermore, on those three text datasets (news20, simacompetition2007 and reuters-21578), the query vectors constructed by SVM_{1c} and SVM_{ba}^{perf} / SVM_{pr}^{perf} are much sparser in average than those constructed by the Centroid algorithm and the Rocchio algorithm respectively. According to one-sided t-test [21, 33], the sparsity superiority of SVM_{1c} over the Centroid algorithm and the sparsity superiority of SVM_{ba}^{perf} / SVM_{pr}^{perf} over the Rocchio algorithm are both at the significance level 99.5% (P_value < 0:005).

In summary, SVM_{1c} works as effectively as the Centroid algorithm while providing much sparser query vectors; both SVM_{ba}^{perf} and SVM_{pr}^{perf} work much better than the Rocchio algorithm, with SVM_{ba}^{perf} being much faster than SVM_{pr}^{perf}.

4. Related Work

PU Learning has been studied recently in the context of text classification [3, 2, 5, 12, 18, 17, 15, 7, 35, 34, 37, 38]. Actually the above mentioned Rocchio algorithm, though simple, has been shown to be quite competitive for PU Learning [37]. Please refer to Liu's new book on Web data mining [16] for a comprehensive survey of this field.

Generally speaking, most existing approaches to PU Learning follow a two-step heuristic: (1) constructing a small reliable negative set $\widehat{N}$ by extracting some examples from *U* which are very unlike positive examples; (2) building a classifier based on *P* and $\widehat{N}$ iteratively. Our principled approach to PU Learning is very different with them, and we study this problem in the context of IR.

The work most related to ours is probably the Biased-SVM method which has shown excellent classification performance in comparison to other state-of-the-art PU Learning methods [17]. It also attempts to optimise a multivariate performance measure (proportional to *pr*), but through an indirect trail-and-error way: it tries a large number of SVM classifiers each with a different cost parameter and then pick one from them according to the classification performance on a held-out validation set. So theoretically Biased-SVM could not work better than SVM_{pr}^{perf}. Our principled approach to PU Learning has several advantages over Biased-SVM:

Approach	Query Vector
Centroid	$\mathbf{w} = \sum_{i=1}^{l} \frac{1}{l}\mathbf{x}_i$
SVM_{1c}	$\mathbf{w} = \sum_{i=1}^{l} \alpha_i \mathbf{x}_i \ (\alpha_i \geq 0)$
Rocchio	$\mathbf{w} = \sum_{i=1}^{l} \frac{1}{l}\mathbf{x}_i - \sum_{j=1}^{u} \frac{1}{u}\mathbf{x}_j$
SVM_{ba}^{perf} etc.	$\mathbf{w} = \sum_{i=1}^{l+u} \alpha_i \mathbf{x}_i \ (\alpha_i \geq 0)$

Table 9. Our proposed approaches to QBME.

- it should be more effective because it *directly* optimises meaningful multivariate performance measures;
- it is hundreds of times more efficient because it only needs to train one SVM classifier;
- a held-out validation set is no longer a prerequisite.

Extensive empirical comparison between SVM_{pr}^{perf} and Biased-SVM is beyond the scope of this paper and left to future work.

There are other fast SVM training algorithms such as [31], though we have not considered how they can be utilized in the QBME setup.

The problem of QBME is different with that of Learning to Rank (LETOR) [19] which has received a lot of attention recently in the IR community. In QBME, a ranking function (i.e., query vector) is learned for each individual query. In contrast, LETOR attempts to learn a query-independent ranking function.

5. Conclusions

In this paper we have proposed to solve the problem of QBME by learning the query vector with SVMs. Specifically we have presented a new method to train one-class SVM in *linear* time as well as a new method to train SVMs from positive and unlabeled examples in linear or *polynomial* time for some meaningful multivariate performance measures (namely balanced accuracy, precision-recall product and precision/recall at *k*).

Our proposed approaches to QBME are summarized in Table 9. Our proposed SVM learning algorithms for QBME are compared in Table 10.

In principle, we can apply the "kernel trick" [27] to the above SVM formulations to achieve non-linear ranking functions, though efficient implementation would be a challenge.

It would be also be interesting to adapt other machine learning algorithms like Voted Perceptron [6] and AdaBoost [25] for QBME.

Learning Algorithm	Performance Measure for optimisation	Training Time Complexity
SVM_{1c} (SVM_{1c}^{struct})	One-Class Accuracy	Linear w.r.t. the size of P, i.e., l
SVM_{ba}^{perf} (SVM_{we}^{struct})	Balanced Accuracy	Linear w.r.t. the size of $P \cup U$, i.e., $l+u$
SVM_{pr}^{perf}	Precision-Recall Product	Polynomial w.r.t. the size of $P \cup U$, i.e., $l+u$
$\text{SVM}_{p@k}^{perf}$ or $\text{SVM}_{r@k}^{perf}$	Precision/Recall at k	

Table 10. Comparison of the SVM Learning algorithms for QBME.

Acknowledgements

We thank the anonymous reviewers for their helpful comments.

References

[1] Blum, A., Mitchell, T (1998). Combining labeled and unlabeled data with co-training. *In Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT),* Madison, WI, p.92–100.

[2] Denis, F., Gilleron, R., Letouzey, F (2005). Learning from positive and unlabeled examples. *Theoretical Computer Science,* 348(1) 70–83.

[3] Denis, F., Gilleron, R., Tommasi, M (2002). Text classification from positive and unlabeled examples. *In Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU),* Annecy, France, p.1927–1934.

[4] Dumais, S., Platt, J., Heckerman, D., Sahami, M (1998). Inductive learning algorithms and representations for text categorization. *In Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM),* Bethesda, MD, p.148–155.

[5] Elkan, C., Noto, K (2008). Learning classifiers from only positive and unlabeled data. *In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD),* Las Vegas, NV, USA, p.213– 220.

[6] Freund, Y., Schapire, R.E (2004). Large margin classification using the perceptron algorithm. *Machine Learning,* 37(3) 277–296.

[7] Fung, G.P.C., Yu, J.X., Lu, H., Yu, P.S (2006). Text classification without negative examples revisit. *IEEE Transaction of Knowledge and Data Engineering (TKDE),* 18(1) 6–20.

[8] Joachims, T (1998). Text categorization with support vector machines: Learning with many relevant features. *In Proceedings of the 10th European Conference on Machine Learning (ECML),* Chemnitz, Germany, p.137–142.

[9] Joachims, T (2005). A support vector method for multivariate performance measures. *In Proceedings of the 22nd International Conference on Machine Learning (ICML),* Bonn, Germany, p.377–384.

[10] Joachims, T (2006). Training linear SVMs in linear time. *In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD),* Philadelphia, PA, p.217–226.

[11] Kelley, J.E. (1960). The cutting plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics (SIAM),* 8, 703–712.

[12] Lee, W.S., Liu, B (2003). Learning with positive and unlabeled examples using weighted logistic regression. *In Proceedings of the 20th International Conference on Machine Learning (ICML),* Washington, DC, p.448–455.

[13] Lewis, D.D., Yang, Y., Rose, T.G., Li, F (2004). RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR),* 5, 361–397.

[14] Li, F., Yang, Y (2003). A loss function analysis for classification methods in text categorization. *In Proceedings of the 20th International Conference on Machine Learning (ICML),* Washington, DC, USA, p.472–479.

[15] Li, X. Liu, B (2003). Learning to classify texts using positive and unlabeled data. *In Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI),* Acapulco, Mexico, p.587–594.

[16] Liu, B (2006). Web Data Mining: Exploring Hyperlinks, Contents and Usage Data. Springer.

[17] Liu, B., Dai,Y., Li, X., Lee, W.S., Yu, P.S (2003). Building text classifiers using positive and unlabeled examples. *In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM),* Melbourne, FL, p.179–188.

[18] Liu, B., Lee, W.S., Yu, P.S., Li, X (2002). Partially supervised classification of text documents. *In Proceedings of the 19th International Conference (ICML),* Sydney, Australia, p.387–394.

[19] Liu, T.Y., Qin, T., Xu, J., Xiong, W., Li, H (2007). LETOR: Benchmark dataset for research on learning to rank for information retrieval. *In Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR).*

[20] Manning, C.D., Raghavan, P., Sch¨utze, H (2008). Introduction to Information Retrieval. Cambridge University Press.

[21] Mitchell, T (1997). Machine Learning. McGraw Hill, International edition.

[22] Radlinski, F., Broder, A.Z., Ciccolo, P., Gabrilovich, E., Josifovski,V., Riedel, L (2008). Optimizing relevance and revenue in ad search: A query substitution approach. *In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR),* Singapore, p.403–410.

[23] Ramakrishnan, R., Gehrke, J (2002). Database Management Systems. McGraw-Hill, 3rd edition.

[24] Salton, G (1971). The SMART Retrieval System. Prentice Hall, Englewood Cliffs, NJ.

[25] Schapire, R.E., Singer, Y (2000). Boostexter: A boostingbased system for text categorization. *Machine Learning,* 39(2/3) 135–168.

[26] Scholkopf, B., Platt, J., Shawe-Taylor, J., Smola, A.J (2001). Estimating the support of a high-dimensional distribution. *Neural Computation,* 13(7) 1443–1471.

[27] Scholkopf, B., Smola, A.J (2002). Learning with Kernels. MIT Press, Cambridge, MA.

[28] Scholkopf, B., Smola, A.J., Williamson, R., Bartlett, P (2000). New support vector algorithms. *Neural Computation,* 12, 1207–1245.

[29] Sebastiani, F (2002). Machine learning in automated text categorization. *ACM Computing Surveys,* 34(1) 1–47.

[30] Sokolova, M., Japkowicz, N., Szpakowicz, S (2006). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. *In Proceedings of the AAAI'06 workshop on Evaluation Methods for Machine Learning.*

[31] Tsang, I.W., Kwok, J.T., Cheung, P.-M. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research (JMLR),* 6, 363– 392.

[32] Tsochantaridis, I., Joachims, T., Hofmann, T., Altun,Y (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR),* 6, 1453–1484.

[33] Yang, Y., Liu, X (1999). A re-examination of text categorization methods. *In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR),* Berkeley, CA, p.42–49.

[34] Yu, H (2005). Single-class classification with mapping convergence. *Machine Learning,* 75(1) 49–69.

[35] Yu, H., Han, J., Chang, K.C.-C. (2004). PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering (TKDE),* 16(1) 70–81.

[36] Zhang, D., Chen, X., Lee, W.S (2005). Text classification with kernels on the multinomial manifold. *In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR),* Salvador, Brazil, p.266–273.

[37] Zhang, D., Lee, W.S (2005). A simple probabilistic approach to learning from positive and unlabeled examples. *In Proceedings of the 5th Annual UK Workshop on Computational Intelligence (UKCI),* London, UK, p.83–87.

[38] Zhang, D., Lee, W. S (2008). Learning classifiers without negative examples: A reduction approach. *In Proceedings of the 3rd IEEE International Conference on Digital Information Management (ICDIM),* London, UK, p.638–643.

[39] Zhang, D., Lee, W. S (2008). Learning with support vector machines for query-by-multiple-examples. *In Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR),* Singapore, p.835–836.