An InterActive Algorithm For Asking And Incorporating Feature Feedback into Support Vector Machines

Hema Raghavan*
Yahoo! Inc
2821 Mission College Blvd.
Santa Clara, CA 95054
raghavan@yahoo-inc.com

James Allan
University of Massachusetts
140 Governor's Drive
Amherst, MA -01003
allan@cs.umass.edu

ABSTRACT

Standard machine learning techniques typically require ample training data in the form of labeled instances. In many situations it may be too tedious or costly to obtain sufficient labeled data for adequate classifier performance. However, in text classification, humans can easily guess the relevance of features, that is, words that are indicative of a topic, thereby enabling the classifier to focus its feature weights more appropriately in the absence of sufficient labeled data. We will describe an algorithm for *tandem learning* that begins with a couple of labeled instances, and then at each iteration recommends features and instances for a human to label. Tandem learning using an "oracle" results in much better performance than learning on only features or only instances. We find that humans can emulate the oracle to an extent that results in performance (accuracy) comparable to that of the oracle. Our unique experimental design helps factor out system error from human error, leading to a better understanding of when and why interactive feature selection works.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*information filtering*; H.1.2 [**Models and Principles**]: user/machine systems—*human factors*

General Terms

Algorithms, Experimentation

Keywords

text classification, term feedback

1. INTRODUCTION

Supervised learning methods like Support Vector Machines that are popularly used in text classification rely on the presence of a large amount of training data that is often procured by having humans (either paid or voluntary) label examples with categories of interest. Companies like Yahoo! employ paid editors to label data and decreasing the amount of labeled data helps to reduce annotation costs. On the other hand, for applications like personalized news or email filtering where a user may be willing to label some data (as little as possible), an algorithm that requires a user to label as little data as possible serves to decrease the tediousness of the process, thereby increasing user satisfaction.

Text classification is a domain in which many irrelevant and redundant features abound and a large amount of training data is typically required for a classifier to automatically discern the relevant features from the irrelevant ones. In text classification, humans can often easily point out relevant features which can help bootstrap the classifier, especially when the number of training examples is few. For example, in classifying documents that are on the topic of *Hurricane Mitch* from those that are not, a human can easily point out that the words *hurricane*, *Mitch* and *Nicaragua* are on topic. A recent paper by Raghavan et al. [18] motivates the usefulness of interactive feature selection for text classification and particularly for active learning when the number of labeled examples are few. Intuitively, human feature selection helps focus the classifier on the more important features, something automatic feature selection techniques cannot do in the absence of sufficient labeled data. In many domains where ample unlabeled data is available, active learning, semi-supervised learning, and transductive learning techniques aim to reduce the amount of labeled data required, by leveraging information from a large pool of unlabeled examples.

In our proposed *tandem learning* algorithm, which is built on top of traditional active learning, a human is asked to label a set of instances and a set of system chosen features, with the objective of learning a classifier with less labeling effort for the human. In Section 3 we will describe an algorithm for *tandem learning* that begins with a couple of labeled instances, and then at each iteration recommends features and instances for a human to label. The algorithm contains methods to incorporate feature feedback into Support Vector Machines.

Aside from a new interactive learning algorithm, an important aspect of this paper is in its experimental design. We use an "oracle" constructed from ample labeled data to explore the benefits of feature selection. The feature oracle is designed to validate the effectiveness of the algorithm. We then compare the oracle performance with true human performance. We prescribe such an oracle approach for any work that involves a human-in-the-loop since it helps separate algorithmic error from human error. In summary, the chief contributions of this work are: (1) An interactive text classification algorithm that recommends terms and documents for a human to label with the aim of decreasing the amount of labeled

*This work was done while the author was at the University of Massachusetts

SIGIR'07, July 23–27, 2007, Amsterdam, The Netherlands.

data required to train a classifier. The proposed algorithm improves the classification performance by 10% (absolute difference, 88% relative difference) over traditional active learning. (2) A preliminary user study that validates users' abilities to label the necessary features. Users overlap significantly with the oracle (over 60%) resulting in performance on par with that of the oracle. We now move on to review related work in machine learning and interactive information retrieval.

2. RELATED WORK

Sebastiani's survey paper [23] provides an overview of techniques in text categorization, a research problem that sits in the joint space of machine learning and information retrieval. In this section we compare our work with past work and their advantages and disadvantages, stating how our work either compares with, or overcomes the deficiencies of past methods.

Our proposed method is an instance of query-based learning [2] and an extension of standard ("pool-based") active learning which focuses on selective sampling of instances from a pool of unlabeled data [6]. To the best of our knowledge, all prior work on query learning and active learning focused on variants of membership queries, that is, requesting the label of an instance. Some recent works have considered using user prior knowledge to bootstrap learning but they typically assume that prior knowledge is given at the outset [27, 21, 8]. The proposed techniques typically involve "soft labeling" instances containing the user labeled features by assigning them to categories associated with those of the features. We expect that our proposed interactive mode has an advantage over requesting prior knowledge from the outset, as it may be easier for the user to identify or recall relevant features while labeling documents in the collection and being presented with candidate features. Some recent work has proposed extending the query model to include feature as well as document level feedback [18]. That work largely demonstrated the need to consider such a dual mode of feedback showing the benefits of such an approach. Their final algorithm was preliminary. Their simple method of scaling the user labeled features is one technique we explore in this paper. Our final algorithm which uses scaling in combination with variants of soft-labeling surpasses any of these individual techniques (scaling or soft labeling). The work of Huang and Mitchell [10] is similar in that a user is queried on features and documents at the same time. However, other than the difference in algorithms, an important aspect of our work is in the experimental set up which helps us better understand the benefits of user term feedback.

User term feedback is common in information retrieval. A significant aspect of this work as compared to other works in interactive information retrieval is in the experimental setup that separates algorithmic error from human error. Most interactive IR experiments have an implicit hypothesis that term feedback will be useful. On this basis, user studies are designed to measure if users can provide feedback without considering whether the algorithm is capable of incorporating feedback in an effective way. More recently the works of Magennis and Rijsbergen [16] and Ruthven [20] try to design experiments for term feedback in the ad-hoc retrieval task in a manner similar to ours. Magennis and Rijsbergen had some deficiencies in their approach, a fact that they acknowledge. Ruthven improved on their experimental setup and found that term feedback using an oracle could give performance improvements even over automatic query expansion. However, he found that users cannot mark the terms required by the optimal query with reasonable precision. Oddly though, he did not report the performance achieved by the user marked terms. In fact, we find that even though users marked only a fraction of the terms marked by the oracle, the performance improvements were on par with the oracle. Besides, it is important to note that our work is in the domain of classification while theirs was in ad-hoc IR.

3. THE TANDEM LEARNING ALGORITHM

In traditional active learning, the learner begins with a few randomly sampled labeled instances. An initial classifier is learned and then active learning begins. Active learning is an iterative, human-in-the-loop learning paradigm in which, at each iteration, the learner picks an instance, such that the knowledge of the label of that instance provides maximum benefit to the learner. A popular active learning algorithm is "uncertainty sampling", in which the human is queried on an instance that the learner is most uncertain about [14].

We extend the traditional active learning framework to engage the human in providing feedback on features in addition to instances. In the active learning literature the human is called a teacher. We use the terms human, teacher and user interchangeably. The outline of the algorithm is shown in Fig.1. The terminology is very similar to the paper by Raghavan et al. [18]. Steps 1 to 3.c are identical to standard active learning system where the learner begins with a few randomly picked instances and then queries the user on examples that it is most uncertain about. X_i and Y_i, denote an instance and its corresponding label. $\mathcal{U}$ denotes the "pool" of unlabeled instances available to the learner to choose queries from. $\mathcal{M}$ denotes the model. The suffix on $\mathcal{M}$ and $\mathcal{U}$ denote their values when a given number of instances have been labeled. We use a semi-batch approach to active learning, where the teacher is queried on I instances in each round of active learning rather than one. A larger value of I implies a greater savings in time, at some cost to effectiveness because the model $\mathcal{M}$ will not be updated after each instance is selected, and the Instance_Selection subroutine will not be able to exploit the label of a previously labeled example X_{t+i} in choosing the subsequent instance X_{t+j} ($t \leq i < j \leq t + I$).

The active learner presents a list $\mathcal{P} = \{P_1 ... P_f\}$ of f features for the teacher to judge (step 3.d) at each iteration. P_i denotes the index of a feature ($1 \leq P_i \leq N$). The teacher is asked to choose one of the following options for each feature, the third choice being the default : (1) Is the feature more likely to occur in *relevant (on-topic)* documents? (2) Is the feature more likely to occur in *non-relevant (off-topic)* documents? (3) Don't know.

Let $\mathcal{F} = \{F_1, ..., F_k\} \subseteq \mathcal{P}$ denote the subset of features marked with either choice (1) or (2) by the user. The simplest implementation of such a system can consist of one where $\mathcal{F}$ is the union of all features in documents in a batch. This idea can be implemented as an interface where the user is asked to highlight relevant words, phrases or passages while reading the document in order to label the document (Step 3.b), akin to the system in the paper by Croft and Das [7]. We prefer to have the set $\mathcal{P}$ to be an ordered one, where feedback on P_1 is more valuable for the learner than feedback on P_2 and so on. The algorithm is called *tandem learning* since the classifier $\mathcal{M}$ is learned on features and documents simultaneously.

4. SUPPORT VECTOR MACHINES

The basic framework for our algorithm is Support Vector Machines [22] since linear SVMs have been shown to be effective for text classification [15] and active learning using SVMs has also been shown to be effective for text [25]. SVMs are linear binary classifiers that maximize the margin of separation between the two classes. Given a training set composed of t examples and the associated class information for each example, the objective is to obtain a hyperplane that best separates the data. More formally, the training set $\mathcal{T}$ consists of t example and class-label pairs ($\mathcal{T} = \{(X_1, Y_1) ... (X_t, Y_t)\}$). Each X_i is represented as a vector, $\{x_{i,1} ...$

Tandem Learning
Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), init_size (number of random feedback iterations), I (Number of instances labeled in each round)
Output: $\mathcal{M}_T$ (Model)
$t = 1; \mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a. $\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_0, \mathcal{U}_{t-1}, random$)
   b. Teacher assigns label Y_t to X_t
   c. $t++$
2. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t-1\}, \mathcal{M}_0$)
3. While $t \leq T$
   a. $\langle X_t, ...X_{t+I-1}, \mathcal{U}_{t+I-1} \rangle$=Instance_Selection($\mathcal{M}_t, \mathcal{U}_{t-1}$, $uncertain$)
   b. Teacher assigns label $Y_t...Y_{t+I-1}$ to $X_t...X_{t+I-1}$
   c. $\mathcal{M}_{t+I}$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t+I-1\}, \mathcal{M}_t$)
   d. i. $\{P_1, ..., P_f\}$ = Feature_Selection($\mathcal{M}_{t+I}, \mathcal{U}_t$)
      ii. Teacher selects $\mathcal{F} = \{F_1, .., F_k\} \subseteq \{P_1, ..., P_f\}$
   e. $\mathcal{M}_{t+I}$=Incorporate_Feature_Feedback($\mathcal{M}_{t+I}$, $\{F_1, ..., F_k\}$)
   f. $t = t + I$
Return $\mathcal{M}_T$.

Figure 1: An active learning system where feedback on features is also requested.

$x_{i,N}\}$, of N features. The classes belong to one of $\{+1, -1\}$ (i.e., $Y_i \in \pm 1$) with $+1$ denoting the label associated with an "on-topic" document and -1 denoting an "off-topic" document. In text classification the vectors are typically L2 normalized.

A hyperplane is given by the pair (w, b), with w being the direction vector of the hyperplane and b the bias. If the data is linearly separable then we can find a pair (w, b) such that:

$$Y_i(w \cdot X_i + b) \geq 1 \; \forall i = 1...t \tag{1}$$

Soft Margin Classifier: In reality all constraints in Equation 1 will not be satisfiable. To account for the violation of constraints, a set of t slack variables $\{\xi_i\}_{i=1}^{t}$, each corresponding to the classification error for a training instance is introduced. The optimization problem involves maximizing the margin and minimizing error.

$$\min_{w,b,\Xi} \phi(w, \Xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{t} \xi_i$$

$$\text{subject to} \quad Y_i(w \cdot X_i + b) \geq 1 - \xi_i \tag{2}$$

$$\xi_i \geq 0 \tag{3}$$

The constant C is user specified and is often referred to as the misclassification cost. A value of 10 is found to be effective for text-classification problems [11]. Solving the above optimization problem is equivalent to minimizing the Lagrangian:

$$L(w, b, \Lambda, \Xi, \Gamma) = \frac{1}{2}||w||^2 - \sum_{i=1}^{t} \lambda_i [Y_i(w \cdot X_i + b) - 1] + \sum_{i=1}^{t} \gamma_i \xi_i + C(\sum_{i}^{t} \xi_i) \tag{4}$$

The predicted class of an instance X_j is then given by $Y'_j = sgn(\sum_{i}^{t} Y_i \lambda_i X_i^T x_j + b)$. The λ_i values denote the extent of influence of a training instance and are constrained to lie between 0 and C for the soft margin classifier. The λ_i values are non-zero only for support vectors (training examples on the margin). For more details on Support Vector Machines we refer the reader to external references (eg., [22]). We can easily specify a different C for different training examples, and in this way control the maximum influence of a given training example. We will exploit this fact in the design of our algorithm.

Uncertainty sampling [14] is a form of active learning in which the example that the user (teacher) is queried on is the unlabeled instance that the classifier is least confident about. When the classifier is an SVM, unlabeled instances closest to the margin are chosen as queries [26]. This is one way of implementing the *Instance_Selection* subroutine in Fig. 1. If an uncertain instance lies exactly on the hyperplane it reduces the version space in exactly half [26]. For many text classification problems uncertainty sampling decreases the number of training examples over random sampling [14].

5. ASKING FOR FEEDBACK ON FEATURES

We now describe our algorithm for implementing the Feature_Selection subroutine in Fig. 1. For a given tandem learning iteration (outer loop in Fig. 1), the Feature_Selection subroutine queries the teacher iteratively presenting features of the set $\mathcal{P}$, in batches. The algorithm is outlined in Fig. 2.

Feature_Selection
Input: $\mathcal{M}_t, \mathcal{U}_t$
1. $\mathcal{S}$ =Extract_top_features($\mathcal{M}_t, p$)
2. $\mathcal{F} = \phi$
3. While ($\mathcal{S} \neq \phi$) and ($B_f \geq 0$)
   a. top=pop($\mathcal{S}$)
   b. $\{P_1, ..., P_c\}$=compute_co_occuring(top,$\mathcal{U}_t$)
   c. Teacher selects $\{F_1, ..., F_{c'}\} \subseteq \{P_1, ..., P_c\}$
   d. push($\mathcal{S}$,$F_1, ...F_{c'}$)
   e. $\mathcal{F} = \mathcal{F} \cup \{F_1, ..., F_{c'}\}$
   f. $B_f = B_f - c$
4. Return $\mathcal{F}, B_f$

Figure 2: An algorithm for Interactive Feature Selection

A budget counter B_f keeps track of the number of features (across tandem learning iterations) that the user has been queried on. A stack of features ($\mathcal{S} = s_1, s_2, ..., s_{|\mathcal{S}|}$) to query the user on is maintained. The size of the stack $\mathcal{S}$ is dynamic as we will see. At each iteration of tandem learning the stack is initialized with the p top ranked features from the current model $\mathcal{M}$, such that the highest ranking feature is at the top of the stack (Step 1 in Fig. 2). The topmost element of the stack ($top = s_1$) is popped at each iteration and the top c co-occuring features to s_1 in the pool ($\mathcal{U}_t$) are computed (Step 3.b). These c features are shown to the user. If the user marks a feature as relevant, it is pushed on top of the stack (Step 3.d). The procedure continues until the stack is empty or the budget B_f is exhausted. Typically $B_f >> c$ and $B_f >> p$. A user may be shown a minimum of 0 features in a given tandem learning iteration if the budget (B_f) is exhausted, and a maximum of B_f features (if we keep finding features greedily). Thus f in the algorithm in Fig. 1 varies across tandem learning iterations. The algorithm is greedy in spirit: when a relevant feature is found, we keep querying the user on words that co-occur with it in the corpus, proceeding as if engaging in a depth first search on a term co-occurence graph. Our early experiments showed that there are benefits of such an approach, especially for the soft labeling method described in Section 6.3.

6. INCORPORATING FEATURE FEEDBACK INTO SUPPORT VECTOR MACHINES

We now move on to describe three methods to incorporate feature feedback into SVMs (step 2.e in the algorithm in Fig. 1).

6.1 Scaling

Let $\mathcal{F} = F_1...F_k$ be the set of features marked relevant by a user, with each F_i representing an index to a feature ($1 \leq F_i \leq N$). Let $Sc = Sc_1...Sc_N$ be a vector such that $Sc_i = a$ if $i \in \mathcal{F}$ and $Sc_i = d$ otherwise. For each X_i in the labeled and unlabeled data sets, we compute the dot product $Sc \cdot X_i$. That is, we scale all the features that the user indicated as relevant by a and the rest of the features by d ($a \geq d$). The documents are re-normalized after scaling. Raghavan et al. [18] use scaling in their interactive feature selection algorithm. It is a simple and effective way of focusing the classifier on user chosen features. This is incidentally equivalent to setting the prior on the weights to have non-uniform variance.

6.2 Feature Pseudo Documents

In this method we create N pseudo-documents, one corresponding to each feature, adding N more N dimensional vectors ($\{\langle X_j, Y_j\rangle, M+1 \leq j \leq M+N\}$) to our already existing unlabeled pool of M document vectors. Thus, a vector X_j corresponding to a feature-id $j-M$ will have a one in the position $j-M$ and zero elsewhere. In this case, the user must associate a class label for every feature that she considers discriminatory. We can now include such feature and feature-label pairs in training the SVM. This method enables us to perform uncertainty sampling with the modified pool.

Initial experiments suggested that there must be a parameter to control the extent of the influence of feature pseudo-documents on the hyper-plane. We can do this in one of two ways. The first method controls the influence of pseudo-document support vectors that are correctly classified, and the second method controls the extent of error tolerated for misclasified pseudo-document support vectors.

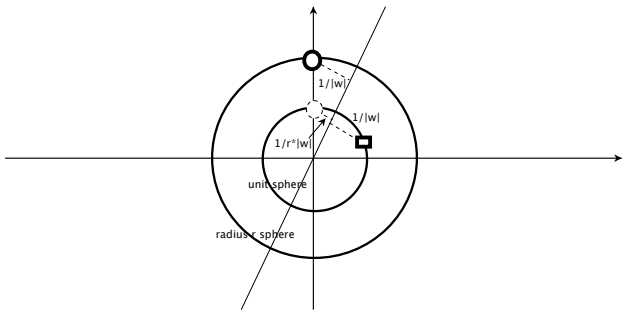

Figure 3: Feature pseudo documents and our modification of the weighted margin classifier.

Method 1: For support vectors that are correctly classified, the distance from the hyperplane is fixed at $1/||w||$. This is forced by imposing the constraints in Equation 1. There are benefits to being able to control this distance for different training examples, letting more reliable instances exert a greater force on the hyperplane We let the feature pseudo documents reside on an r-radius hypersphere instead of the unit hypersphere that the documents reside on. A two-dimensional example is shown in Fig. 3. Let X_j be a feature pseudo document on an r-radius hypersphere. The feature pseudo-document X_j ($\forall\, j = M+1...M+N$) corresponding to a feature $j-M$ now needs to have a value r in position $j-M$ and 0 elsewhere. This support vector is forced to be at distance $1/||w||$ from the hyperplane. Projecting down to the unit hypershere (on which the documents lie), the distance is $1/(r \times ||w||)$. Hence, the feature pseudo-documents are forced to be at a distance of $1/(r \times ||w||)$ from the hyperplane on the unit hypersphere. Meanwhile the document support vectors continue to lie at distance $1/||w||$ from the hyperplane on the unit hypersphere. If $r \leq 1$ the feature pseudo document support vectors exerts a greater influence on the margin than the document support vectors. The implementation is very simple and the same QP solver used for the soft margin SVM can be used to find the solution.

The idea to use feature pseudo documents first appeared in a paper by Godbole et al. [9]. Our implementation differs in that we introduce parameters to influence the extent of control of the feature pseudo-documents on the hyper-plane as compared to the extent of influence of the training documents. The idea to weight different instances differently appeared in a paper by Wu and Srihari but their implementation changes the optimization problem while ours is simpler [27]. Also note that Wu and Srihari did not use feature pseudo-documents, and instead used the instance weighting technique to weight documents.

Method 2: The second method controls the extent of influence of the misclassified support vectors. We can do this by controlling the weight C for the feature pseudo documents by modifying the optimization problem as follows:

$$\begin{aligned} \min_{w,b,\Xi} \phi(w,\Xi) &= \frac{1}{2}||w||^2 + C\sum_{i=1}^{t}\xi_i + \frac{C}{|\mathcal{F}|}\sum_{i=t+1}^{t+|\mathcal{F}|}\xi_i \\ \text{subject to} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i \ , \ \xi_i \geq 0 \end{aligned} \quad (5)$$

where $|\mathcal{F}|$ is the number of terms a user has marked, and also equals the number of training pseudo-documents. The upper bound on the λ_i values corresponding to the training pseudo-documents ($t < i \leq t + |\mathcal{F}|$) is $C/|\mathcal{F}|$. In this way, as more training pseudo documents are available (probably due to the topic being very descriptive), the influence of an individual pseudo-document vector is decreased. Note that λ_i values for the training documents ($1 \leq i \leq t$) continue to remain bounded by C (where $C \geq C/|\mathcal{F}|$).

6.3 Pseudo Relevance Feedback

We now consider soft labeling the unlabeled examples in $\mathcal{U}$ and using them in the training. The assumption here is that unlabeled examples containing a term that the user has associated with a given class are likely to belong to that class, thereby enabling us to assign a "soft-label" to the document. The greater the number of terms that the user has marked for a given class that appear in a document, the greater the confidence in our soft-label. It is easy to see why this method benefits from the labeling of redundant features.

Let $\mathcal{F} = \mathcal{F}^+ + \mathcal{F}^-$ where $\mathcal{F}^+$ and $\mathcal{F}^-$ denote the set of terms that the user has associated with the classes corresponding to the labels $+1$ and -1 respectively. Let v_i+ denote the similarity of an unlabeled document X_i ($X_i \in \mathcal{U}$) to $\mathcal{F}^+$. Similarly we can compute v_i^-. v_i can denote any similarity metric of choice. Let $g(v_i)$ be a monotonically increasing function with range $(0, 1]$. We now modify the optimization problem to include unlabeled instances as follows:

Category	Terms that are Support Vectors
earn	qtr, note
acquisitions	qtr, ct, shr, ct_net, mln_mln, ct_ct;
money-fx	-
crude	crude
trade	japan
interest	bank
wheat	export, maize
corn	-
money-supply	bank, dlr
gold	mine, gold_mine

Table 1: Feature pseudo documents that are support vectors (Reuters 21578).

$$\min_{w,b,\Xi} \phi(w,\Xi) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{t}\xi_i + C\sum_{i=t+1}^{t+|\mathcal{U}|} g(v_i^+)\xi_i + C\sum_{i=1}^{t+|\mathcal{U}|} g(v_i^-)\xi_i$$
$$\text{subject to} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i\,, \xi_i \geq 0 \qquad (6)$$

This idea has been used successfully in the past [27]. Like in that work, v_i is simply set to be the cosine similarity and $g(v_i) = v_i$ since v_i has the desired $(0, 1]$ range.

7. EXPERIMENTAL SETUP

We used a total of four corpora for our experiments. The Reuters-21578 corpus [19], a standard benchmark for text categorization, was used as the development data set. The remaining corpora formed our test sets. The second corpus is the 20-Newsgroups dataset collected by Lang[13] with about 20,000 documents which are postings on 20 Usenet newsgroups. The third corpus is the TDT3 corpus [1] provided by the LDC that has documents in multiple languages from both broadcast and news-wire sources. Broadcast output is available as the output of an automatic speech recognizer (ASR). Similarly for documents not originally in English the output of a machine translation (MT) is provided (target language English). We also used the larger reuters RCV1 corpus consisting of about 800K documents[15]. A pool size of 1000 was used for all corpora except the RCV1 corpus where the pool consisted of the training documents in the Mod-Apte split[15] (about 23K documents).

The (feature) oracle in our experiments has access to the labels of all documents in the data-set and uses this information to return a ranked list of features sorted in decreasing order of importance. Information gain is a common measure for ranking features and has been found to be quite effective [23, 4], and is easy and quick to compute. In our oracle experiments we cut off the ranked list at a threshold determined by the shape of the information gain curve in the following way. We took the top 30 features computed by information gain, and computed the average score. All features with scores above the average were considered "relevant" by the oracle. The oracle also associated a category label with a feature by computing the probability of occurrence of a feature in each of the two classes using all the labeled data. The feature was labeled with the category with greater likelihood of containing the feature. Performance using the oracle is a loose upper bound on the effectiveness of feature selection and subject to the accuracy of the feature selection algorithm, but since we use a large amount of data to estimate the oracle, we expect this bound to be fairly tight.

Since accuracy is inappropriate as a measure of effectiveness for many of the data sets as the proportion of positives is very small, we measured performance using the F1 score, the harmonic mean of precision and recall. We measure performance of tandem learning after 10 actively picked documents have been labeled since it has been found that there is little use of feature labeling after this point [18]. Note that a total of 12 documents would have been labeled at this point including the 2 original ($T = 12$). We compare the performance with traditional document only and term only methods. We also resorted to asking the user for 2 documents for feedback at each iteration ($I = 2$). Preliminary experiments revealed that there was little negative impact to accuracy in this approach (as compared to $I = 1$, with some gain in efficiency.

We set the feature feedback quota, B_f, to a value of 100. It has been estimated that a document takes 5 times more time to label than a feature [18]. Their experiment was very conservative, more of a worst case scenario, and in reality one expects feature labeling to be faster, depending on the interface in which it is shown to a user. However, using this upper bound, labeling a 100 features roughly equals the effort needed to label 20 documents. Hence we also compare tandem learning performance at $T = 12$ and $B_f = 100$ with the performance of traditional active learning after 32 documents that is, when $T = 32$ and $B_f = 0$, since labeling 12 documents +100 terms $\preceq$ 12 documents + 20 documents.

In the next section we use an oracle to determine which of the three feature incorporation methods is better. We find a combined approach using all three methods to be best, which is what we use for the final implementation.

System			Macro-F1
Baselines	topdocs + uncertain		0.420
	active		0.516
Tandem learning: feature incorporation techniques (combinations)			
I: Scaling	II: Features as Pseudo-Docs	III: Pseudo-Rel. Feedback	
X			0.553
	X		*0.433*
		X	**0.592**
X	X		0.577
	X	X	**0.597**
X		X	**0.638**
X	X	X	**0.651**

Table 2: Ablation experiments to determine which method of feature feedback is most effective. The figures in bold indicate a significant improvement in performance over the best baseline (0.516). Numbers in italics indicate a significant decrease in performance over the baseline. In all cases $T = 12$.

8. CHOICE OF METHODS

For instance selection we experimented with using 2 uncertain or 2 of the most relevant documents for feedback. We found that for the tandem learning algorithm a mixture of the most relevant and uncertain documents was best. Performance of tandem learning is compared to baseline systems that use both, only actively sampled documents and a mixture of active and top ranking documents for feedback. Initial experiments showed that using only top ranking documents for feedback performed very poorly for classification.

We conducted experiments by simulation using the oracle. Every time a feature was presented to the user for feedback, we labeled it based on the oracle's judgment of the feature. Results of these upper bound experiments are shown in Table 2. Rows two and three give the performance of the two baselines. Uncertainty sampling is clearly better than using a mixture of top ranking and uncertain

documents. We tuned our parameters on a handful of topics in the Reuters 21578 corpus for a system that uses all three modes of feedback together. The results of that experiment are tabulated in row 5. There is a 27% improvement in performance. We then conducted ablation experiments to study the benefit of each feature incorporation method separately (rows 6, 7 and 8) and various combinations of them (rows 9, 10 and 11). Pseudo relevance feedback (Method III) is the best performing method, and by itself gives a significant [1] improvement in performance. Scaling (Method I) on its own, also improves performance, but Method II only slightly improves performance. Method II in conjunction with each of Method I and Method III improves performance (although almost negligibly) over each of these methods individually. A combination of Methods I and III is significantly much better than the baseline, and when Method II is combined with them there is a tiny improvement over that combination. The result of using only active documents for feedback in combination with all 3 feature incorporation techniques is 0.661 (almost identical to the 0.651 value obtained using a mixture of uncertain and top ranking documents). Hence in our final implementation we use a joint approach with all three methods with a combination of active learning and topdocs. Table 1 shows example pseudo documents that are support vectors in the final system. The experimental setup for our final implementation that uses all three methods is described next.

The parameters p, c, a, d and r were set to values of 25, 10, 10, 1 and 10 respectively. The average performance for each topic was computed using 10 different random initializations of the initial 2 documents. We also took care that if the system had already queried the user (or oracle) for a feature in a given iteration, that feature was not asked about in subsequent iterations.

9. RESULTS WITH AN ORACLE

Results on all corpora are tabulated in Table 3. We compare our results with many baselines. The "only documents" methods in rows 2, 3, 4 and 5 do not use user feature feedback at all. The second row shows the results of querying the user on one top ranked document, and one uncertain document in each round of active learning ($init\_size = 2, T = 12, I = 2$). The setup for the third row is identical to the second, except that $T = 32$. The fourth and fifth rows show the results when both documents that the user is asked for feedback on are uncertain ones, for the cases when $T = 12$ and $T = 32$. For most corpora it seems that pure uncertainty sampling is better than a combination of uncertainty sampling and topdocs when $T = 12$ (compare rows 2 and 4) and even when $T = 32$ (rows 3 and 5).

The sixth row shows the results of using the terms marked relevant by the oracle, and no document feedback. For that experiment, we took all terms in this set and issued them as a query to the pool $\mathcal{U}$. A TF-IDF model was used, and the documents were ranked by similarity to the set of oracle marked terms. The top 10 documents were treated as positive documents and the bottom 10 as negative.

The seventh row corresponds to the case when the initial classifier is learned as before ($init\_size = 2$) and subsequently during active learning, the human is queried on only features. The eighth row is the result of using the full tandem learning system. Given that with 2 labeled documents the performance of the classifier is 0.179, 0.154, 0.053 and 0.078 respectively for each of the four corpora, we observe that we are able to recommend enough useful features to improve over the initial classifier by 10% (absolute difference between row 8 and row 5 of the performance of the 20NG, TDT3 and RCV1 corpora).

[1] All significance tests in this paper are paired two-tailed t-tests at the 95% level of confidence.

Comparing the results of tandem learning (eighth row) to methods that use only documents for feedback (rows two through five) we see that for all corpora, feature feedback significantly outperforms a system that uses only 12 documents for training. Tandem learning is also better than using 32 documents for feedback for three of four corpora. Remember 32 documents is a loose upper bound on the effort required to label a 100 features. For the RCV1 corpus, labeling 32 documents results in better performance than tandem learning. However, the improvement in performance of tandem learning for RCV1 over the case when $T = 12, B_f = 0$ (row 3) is much more in magnitude than the loss in effectiveness due to expending extra effort in feature feedback instead of document feedback (compare row 8 and row 5 for RCV1), indicating that the terms labeled are quite useful, and probably only a little less useful than the documents. Note that these results depend on how good the estimate of the oracle is. Many problems in RCV1 have very few positive documents. It is possible that a better estimate of the oracle, say by using domain knowledge will boost performance at least up to that of the case when $T = 32$. 50% 25%, 86% and 98% of the topics in each of the four corpora are improved over the baseline corresponding to row 4 of the table. The topics that were improved for the Reuters and 20 Newsgroups corpora saw substantial improvements while the topics that were hurt suffered negligible change in performance.

Such a comparison, asking how much effort is expended on marking features and whether that effort is better spent in marking documents differentiates our work from all past work [27, 9, 8] in machine learning that uses user prior knowledge to boot-strap learning. From our observations, spending some effort to mark features is almost never an effort wasted. Many times it boosts learning, improving over a paradigm that just uses documents for feedback, and does not hurt performance. Features are also not sufficient in themselves (compare row 8 with rows 6 and 7).

All experiments with feature feedback imposed a quota of $B_f = 100$. The effect of different values of B_f is shown below:

B_f	F1	
	Reuters	20 NG
0	0.420	0.081
10	**0.519**	**0.158**
20	**0.584**	**0.214**
40	**0.634**	**0.282**
80	0.631	**0.361**
100	0.651	0.354

Numbers in bold indicate statistically significant improvements in performance over the previous row. It is well known that the categories in the Reuters corpus are defined by a small set of well selected features, whereas 20 NG needs many more features [3] which explains why 20NG needs a greater feedback quota.

10. A PRELIMINARY USER STUDY

We now ask whether humans can label features that benefit active learning (i.e., the oracle marked features). We also question whether this feedback is beneficial to active learning even if the feature labels people provide are imperfect.

10.1 Experimental Setup

We obtain feedback on features offline, using a TREC "pooling-like" approach and then use the judgments so obtained to measure the effectiveness of user marked features on the tandem learning system. We employed this methodology rather than one where an actual user labels features and documents in tandem because our approach is cheaper and allows us to run many repeated trials of our experiments, also enabling us to do significance testing. We used

		Reuters	20 NG	TDT3	RCV1	Feedback
only documents	topdocs + uncertain	0.420	0.085	0.176	0.081	$T = 12, B_f = 0$
		0.562	0.157	0.153	0.145	$T = 32, B_f = 0$
	uncertain only	0.516	0.180	0.166	0.134	$T = 12, B_f = 0$
		0.570	0.297	0.259	0.260	$T = 32, B_f = 0$
terms w/ docs	only terms (apriori)	0.536	0.340	0.085	0.229	$T = 0, B_f = 0^2$
	iterative terms	0.573	0.335	0.168	0.099	$T = 2, B_f = 100$
	tandem learning	**0.651**	**0.354**	**0.336**	**0.231**	$T = 12, B_f = 100$

Table 3: Results of Tandem Learning with an Oracle. Numbers in bold indicate statistically significant difference in performance over the case when $T = 32$ and $B_f = 0$ with active sampling (line 5). Tandem learning performs less than the $T = 32, B_f = 0$. Tandem learning is always better than the "only document" methods when $T = 12$.

User	Ability to mark features								Effectiveness	
	Kappa				P & R				F1	
	3 class		2 class		P+	P-	R+	R-		
	User 2	Oracle	User 2	Oracle					2 class	3 class
User 1	0.275	0.147	0.569	0.305	0.402	0	0.789	0.900	0.316	0.297
User 2	-	0.350	-	0.359	0.565	0.883	0.649	0.900	0.286	0.287

Table 4: Inter-annotator agreement and performance using the user labeled features on the TDT3 corpus. P+ and R+ denote the precision and recall of the features the user labeled in the "relevant (on-topic)" class with respect to the features that the oracle ascribed to the "relevant class". P- and R- denote the corresponding numbers for the "non-relevant" class. Performance (F1) of tandem learning using user labeled features is comparable to that of the oracle performance of 0.336, and is significantly better than the baseline of 0.176 (corresponding to row 2 of Table 3). The 0.316 performance obtained using User 1's positively labeled features is statistically indistinguishable from the performance of the oracle.

the LDC judgments on documents, and reserved our annotators efforts only for feature judgments. The annotator did not have to judge a given feature for a given problem more than once. We also discarded features that were rarely asked by any algorithm, saving annotator effort significantly. We could also repeat experiments as simulations with our database of relevance judgments allowing us to further develop algorithms for interactive feature selection, testing them using this simulation-like approach.

We ran the system for 10 different choices of the initial positive and negative documents (Step 1 in Fig. 1) for each topic in the TDT3 corpus. Let us call each initialization for a given topic a "run". We concatenated all recommended terms (Step 3.a) keeping track of how many times a term was recommended across different runs. We discarded all terms that were recommended only once across all runs for a topic. For each topic we then added the terms determined relevant by the oracle into the pool giving us an average of 130 terms per topic.

We had one paid annotator judge 60 topics in the TDT3 corpus. She was not a computer scientist, rather a political science major, computer literate and familiar with some basic statistics. She was briefly explained the task in a 15 minute training session. She was also given written instructions. She was given a brief description with access (as a hyper-link) to a detailed topic description before she began making terms. Both the brief and detailed topic descriptions are provided by the LDC.

We sorted all terms alphabetically, and showed her each term with 4 contexts in which it appears in the corpus. A small pane with a reminder of the topic (the brief topic description) and a link to the more detailed description was available to the user at all times. We retrieved context for each feature by issuing that term as a one word query using the Indri toolkit [24]. Context helps disambiguate a word and also provides a snapshot of the senses it appears in, in the corpus. For example context is useful in helping a user determine the meaning of the feature *ct* as *cents* and not *Connecticut* in Reuters documents. The term *ct* is a relevant feature for the Reuters *earnings* category. Another example where context proves to be particularly important is in having the human overcome the effects of machine translation and ASR errors. The word "nobel" is consistently mis-translated as "promises bell" in machine translation documents and hence the word *bell* turns out to be a useful feature for the topic *Nobel prizes awarded.* A user might be able to notice such an error and mark the word "bell" as associated with the relevant documents.

Terms were shown one at a time and at each instance the user was asked to mark one of the three choices in Section 3. We imagine that in a more realistic implementation, terms will be shown as lists, which is probably faster than having a user judge one feature at a time. The term-at-a-time method is however, the best interface for a controlled experiment to measure users' abilities to mark features. In fact in an earlier study we did not even show context and the user was given a very brief topic description. One option for this study was to show an initial screen with a list of features, and for each feature a hyper-link to a context, in case the annotator needed clarification about the meaning of a feature. We avoided this interface because it is possible that a user might not click on a link because of a pre-conceived notion of what the feature means. For example, a user who assumed *ct* stood for Connecticut and never imagined it could mean anything else (cents in our case) would not click on the link.

The author of this paper also judged terms for all 60 topics. Although one might think that the author would represent a biased user, with domain knowledge of the corpus, the underlying algorithm and so on, surprisingly it turns out that she and the paid annotator perform almost on par, especially in terms of the final effectiveness of the tandem learning algorithm as we will see next.

10.2 Results

We measure the extent to which our users tend to agree with each other about the importance of features using the kappa statistic [5], a measure that quantifies the agreement between annotators that independently classify a set of entities (in our case the features) into classes (relevant versus non-relevant/don't know).Table 4 shows the kappa values for each of the two users for the 3-way classification problem (columns 2 and 3). The agreement between User 1 (the paid annotator) and the oracle is "poor", but the agree-

ments between User 1 and User 2 (the author) and between User 2 and the oracle are "fair" (according to the interpretation of Kappa by Landis and Koch[12]). Upon investigation we found that User 1 had a tendency to attribute many features to the "non-relevant" category. User 2, on the other hand typically marked only "relevant" features, leaving all others to the default "don't know" category. This tends to match with the oracle to quite an extent. The oracle marked 12 features (on average over the 60 topics) as belonging to the "relevant" class and 0.013 features as belonging to the "non-relevant". If we collapse, the "non-relevant" and "don't know" categories into one, giving a 2-way classification problem (columns 4 and 5), we see increased agreement between the two users, reflecting an overall "moderate" (and bordering on "substantial") agreement.

We also report precision and recall of each of the users with respect to the oracle in Table 4. As mentioned earlier, the oracle had extracted about 12 terms on average per topic. Users tend to be more verbose than the oracle, with User 1 judging 25 terms (average) in the "relevant class", 29 terms in the "non-relevant" class, and the remaining terms in the "don't know" class. These numbers are 15, and 0.33 respectively for User 2. Both users have very high recall, but relatively lower numbers of precision. Remember that the oracle is constructed from a feature selection algorithm, which might suppress redundant features, whereas the users did not do so. Ultimately it should be the recall with respect to the oracle that matters for effectiveness. In fact it is indeed the case that performance of the algorithm using user labeled features is almost on par with that of the oracle (see last two columns of table 4).

11. CONCLUSIONS

In summary our conclusions are: 1. That there exist a set of features, for which if the learner is provided relevance information, the speed of active learning can be significantly improved. 2. That the learner can pick these features to ask the teacher for feedback on, in a tandem learning like framework. 3. And that these features can be marked by humans fairly easily.

12. FUTURE WORK

Our algorithmic methodology allows us to clearly separate what the system needs to know and can achieve in the limit, and what users are able to provide. Researchers with an engineering focus should aim to better this upper bound performance. And those with an HCI focus should aim at designing interfaces that enable users to achieve the same recall as the oracle. Additional analysis, experiments and ideas for future work can be found elsewhere [17].

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the sponsor.

13. REFERENCES

[1] J. Allan. *Topic detection and tracking*. Kluwer, 2002.

[2] D. Angluin. Computational learning theory: survey and selected bibliography. In *STOC*, pages 351–369, 1992.

[3] R. Bekkerman, R. El-Yaniv, Y. Winter, and N. Tishby. On feature distributional clustering for text categorization. In *SIGIR 01*, pages 146–153, 2001.

[4] J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Feature selection using linear support vector machines. Technical report, Microsoft Research, 2002.

[5] J. Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20, 1960.

[6] D. A. Cohn, L. Atlas, and R. E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

[7] W. B. Croft and R. Das. Experiments with query acquisition and use in document retrieval systems. In *SIGIR '90*, pages 349–368, 1990.

[8] A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *SIGIR '06*, pages 493–500, New York, NY, USA, 2006. ACM Press.

[9] S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. Document classification through interactive supervision of document and term labels. In *PKDD 04*, pages 185–196.

[10] Y. Huang and T. M. Mitchell. Text clustering with extended user feedback. In *SIGIR '06*, pages 413–420, 2006.

[11] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *ECML 98*, pages 137–142, 1998.

[12] G. Landis and G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33:159–174, 1977.

[13] K. Lang. Newsweeder: Learning to filter netnews. In *ICML 95*, pages 331–339, 1995.

[14] D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ICML 94*, pages 148–156, 1994.

[15] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397, 2004.

[16] M. Magennis and C. J. van Rijsbergen. The potential and actual effectiveness of interactive query expansion. In *SIGIR '97*, pages 324–332, 1997.

[17] H. Raghavan. *Tandem Learning: A New Learning Framework for Document Categorization*. PhD thesis, University of Massachusetts, 2007.

[18] H. Raghavan, O. Madani, and R. Jones. Active learning on both features and documents. *JMLR*, 7:1655–1686, 2006.

[19] T. G. Rose, M. Stevenson, and M. Whitehead. The reuters corpus vol. 1 - from yesterday's news to tomorrow's language resources. In *Proceedings of International Conference on Language Resources and Evaluation*, 2002.

[20] I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *SIGIR '03*, pages 213–220.

[21] R. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *ICML 02*.

[22] Scholkopf and Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002.

[23] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surverys*, 2002.

[24] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *International Conference on Intelligence Analysis*, 2005.

[25] S. Tong. *Active learning: theory and applications*. PhD thesis, Stanford University, 2001.

[26] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, 2002.

[27] X. Wu and R. Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *KDD 04*, pages 326–333, 2004.