End-User Feature Labeling: A Locally-Weighted Regression Approach

Weng-Keen Wong[1], Ian Oberst[1], Shubhomoy Das[1], Travis Moore[1], Simone Stumpf[2], Kevin McIntosh[1], Margaret Burnett[1]

[1]Oregon State University
Corvallis, OR 97331 USA
{wong,obersti,dassh,moortrav,mcintoke,burnett}
@eecs.oregonstate.edu

[2]City University, London
London, UK
Simone.Stumpf.1@city.ac.uk

ABSTRACT
When intelligent interfaces, such as intelligent desktop assistants, email classifiers, and recommender systems, customize themselves to a particular end user, such customizations can decrease productivity and increase frustration due to inaccurate predictions—especially in early stages, when training data is limited. The end user can improve the learning algorithm by tediously labeling a substantial amount of additional training data, but this takes time and is too ad hoc to target a particular area of inaccuracy. To solve this problem, we propose a new learning algorithm based on locally weighted regression for *feature labeling* by end users, enabling them to point out which features are important for a class, rather than provide new training instances. In our user study, the first allowing ordinary end users to freely choose features to label directly from text documents, our algorithm was both more effective than others at leveraging end users' feature labels to improve the learning algorithm, and more robust to real users' noisy feature labels. These results strongly suggest that allowing users to freely choose features to label is a promising method for allowing end users to improve learning algorithms effectively.

Author Keywords
Feature labeling, locally weighted logistic regression, machine learning, intelligent interfaces.

ACM Classification Keywords
I.2.6 Artificial Intelligence: Learning, H.1.2. User/Machine Systems: Human factors.

General Terms
Algorithms, Human Factors.

IUI 2011, February 13-16, 2011, Palo Alto, California, USA.

1. INTRODUCTION

Many applications, powered by machine learning, customize themselves to a particular end user's preferences. Such applications include email classifiers, recommender systems, intelligent desktop assistants, and other intelligent user interfaces. To accomplish this customization, the application must learn from the particular end user—which obviously cannot happen until *after* the system is deployed and training data from that specific end user is obtained.

Customizing to the end user's preferences is challenging, especially when there is limited training data, such as when the application is first deployed. The end user could select additional training instances to label, or the learning algorithm could ask the user to provide class labels for strategically chosen instances that would most inform the learning algorithm, as is done in traditional active learning [3]. Labeling instances, however, has its drawbacks. Firstly, labeling data instances is a tedious process and a substantial number of instances must often be labeled before a change to the learning algorithm is noticeable to an end user. Secondly, if a rare group of instances is incorrectly classified, the learning algorithm cannot be "corrected" until the user labels instances with this rare combination of attributes. Since this group is rare, there could be a long wait for enough of these data instances to arrive.

To overcome these problems, in this paper we investigate the possibility of end-user *feature labeling* [18], namely allowing end users to label features instead of instances. The term *feature* refers to an attribute of a data instance that is useful for predicting the class label. For example, rather than labeling entire documents, an end user could point out which words (features) in the document are most indicative of certain class labels, such as in our formative research's user interface shown in Figure 1 [8], which allowed HCI researchers to point out words that were predictive of that transcript segment's label. Raghavan et al. [15, 16] found that labeling a feature took roughly a fifth of the time to label than a document and the benefits of feature labeling were greatest when the training set sizes were small. How-

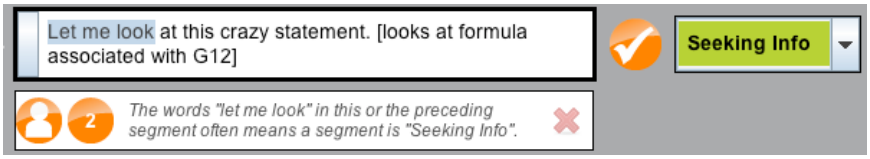


Figure 1: The user is pointing out that the feature "let me look" is highly indicative of the class "Seeking Info." (This UI inspired the development of the algorithm we present in this paper.)

ever, their work did not statistically evaluate feature labeling when performed by actual end users.

Allowing end users, who are not likely to be educated in machine learning, to use feature labeling introduces new challenges to learning algorithms. End users' choices of features may be noisy, inconsistent, and might vary greatly in ability to improve the predictive power of the machine learning algorithm. This paper therefore investigates algorithms able to stand up to these challenges.

Our research contributions are as follows. First, we present a new algorithm for taking end-user feature labels into account, based on Locally Weighted Logistic Regression. Second, we present an empirical comparison on multiple data sets under ideal conditions, using feature labels obtained from an oracle. For these data sets, our algorithm either outperformed or matched the performance of other competitive algorithms. Third, we present the first user study of its kind in this area, in which ordinary end users, unfamiliar with machine learning, chose the feature labels themselves—with no restrictions as to what they could select as features. Using these feature labels provided by ordinary end users, our algorithm again outperformed other algorithms with feature labeling, and was also more robust to poor quality feature labels. We also investigated the characteristics of end users' features to inform the design of interactive feature labeling systems. Finally, we performed a sensitivity analysis and showed that the performance of our algorithm is more robust to different parameter settings than other algorithms. Together, our results strongly suggest that feature labeling by end users is both viable and an effective solution for allowing end users to improve the learning algorithm behind their customized user interface.

2. RELATED WORK

There are a variety of approaches to take feature labels into account. Raghavan and Allen [16] present three methods to deal with feedback on features. We will describe their first two methods in detail since we evaluated our algorithm against them. Method 1 scales features indicated as relevant by the user by a constant a and the rest of the features by d (where $a \geq d$). In Method 2, the user indicates that the jth feature is relevant for a class label l. For each feature-label pair, Method 2 creates a pseudo-document consisting of a value r in index j, zeroes elsewhere, and a class label of l. The r parameter controls the influence of the support vectors of the pseudo-documents on the separating hyperplane.

Another class of techniques for incorporating feature labels takes a semi-supervised approach and leverages information from a large pool of unlabeled data. For instance, Method 3 of [16] soft-labels an unlabeled pool of data using feature-label pairs provided by the user. These soft-labeled documents are associated with slack variables which are then introduced into a modified SVM objective function. Stumpf et al. [20] also use a soft-labeling approach to incorporating feature feedback through their user-cotraining algorithm. In our work, we do not assume that we have a large pool of unlabeled data. Instead, we want to identify the gains from feature labeling when only the labeled instances in the training set are available.

Feature labeling can also be framed in terms of Generalized Expectation (GE) [5], which is a framework for incorporating preferences about variable expectations during parameter estimation. GE is a very general framework that encompasses a wide variety of methods including maximum entropy models, semi-supervised learning and transfer learning [13]. The GE-based feature labeling algorithms in [5] all leverage information from a pool of unlabeled data. Our algorithm can be considered a special case of GE that is different from the previous GE-based algorithms in [5].

Dual supervision [18] is a term used to describe the process of labeling both instances and features. Raghavan and Allen [16] combine their feature labeling with uncertainty sampling for instance labeling in their *tandem learning* approach. Other dual supervision approaches include a graph-based transduction algorithm [18] and an approach using pooled multinomials [1]. The focus of these last two papers is on active learning for dual supervision, which chooses instances *and* labels for labeling. Our work differs in that it is the end users, not the active learning algorithm, that chooses the features for labeling. Furthermore, we are investigating the effects of labeling only features, not instances, especially with an eye to the initial training period when training data is limited.

All of the above methods deal with labeling *existing* features. Roth and Small [17] allow users to create new features by replacing features corresponding to semantically related words with a Semantically Related Word List (SWRL) feature. Their focus, however, was on creating SWRLs to improve classifiers rather than feature labeling. In our user study, we allow end users to construct new features and label them.

Finally, almost all of the prior work in feature labeling evaluates algorithms under ideal conditions, such as feature labels obtained from an oracle [1, 15, 18]. Our study investigates both the use of ideal oracle feature labels and feature labels provided by real end users.

Figure 2: (Left) The Logistic Function fit to two classes: squares (y=0) and circles (y=1). (Middle) A non-separable case where (global) Logistic Regression will have difficulty separating the circle class (y=1) from the square class (y=0), resulting in a poor fit. Note that the square data points to the right will be classified as circles. (Right) A non-separable case where Locally Weighted Logistic Regression will be effective in separating the circle class from the square class.

3. ALGORITHM

Our approach, termed LWLR-FL, incorporates feature labeling into Locally Weighted Logistic Regression (LWLR). To provide context for LWLR-FL. we first describe (global) Logistic Regression and LWLR.

3.1 Background: Locally Weighted Logistic Regression

Logistic Regression (LR) [6] is a well-known method in statistics for predicting a discrete class label y_i given a data instance $\boldsymbol{x_i}=(x_i^1,\ldots,x_i^D)$ with D features; we refer to the dth feature, without reference to a specific data instance, using the superscript notation ie. x^d. LR models the conditional probability $P(y_i|\boldsymbol{x_i})$ by fitting a logistic function to the training data. Figure 2 (Left) illustrates the s-shaped logistic function fit to training data from two classes (squares and circles). Notice that the data is perfectly separable, in the sense that data to the left of the bend in the "S" is classified as a square and data to the right as a circle.

The conditional probability for an M-class problem is:

$$P_{\boldsymbol{\theta}}(y_i = c_j \mid \boldsymbol{x_i}) = \frac{\exp(\beta_j^0 + \sum_{d=1}^{D} \beta_j^d \boldsymbol{x_i^d})}{\sum_{m=1}^{M} \exp(\beta_m^0 + \sum_{d=1}^{D} \beta_m^d \boldsymbol{x_i^d})}$$

In the equation above, the notation c_j refers to the jth class. The parameters $\boldsymbol{\theta} = (\boldsymbol{\beta_0}, \ldots, \boldsymbol{\beta_D})$ are computed by maximizing the conditional log likelihood, which cannot be solved in closed form but must be done numerically.

LR assumes that the parameters $\boldsymbol{\theta}$ are the same across all data points. Although this approach works reasonably well when the classes are linearly well separated, it fails when the actual decision boundaries are more complex and when the data is noisy [4], which is often the case with real-world data. For instance, Figure 2 (middle) illustrates a problematic case for LR when the data is not cleanly separable by the logistic function. Here, the s-shaped logistic function fits the data poorly, resulting in the two squares on the right to be classified as circles.

One solution for dealing with the difficult case in Figure 2 is to use Locally Weighted Logistic Regression (LWLR) [2, 4], in which the logistic function is fit locally to a small neighborhood around a query point $\boldsymbol{x_q}$ to be classified. Figure 2 (right) illustrates LWLR fit to the data points. Intuitively, LWLR gives more weight to training points that are "closer" to the query point than those farther away. A common distance function used to determine the closeness of text documents is cosine similarity. Since we want the distance to increase when a training instance $\boldsymbol{x_i}$ is less similar than the query instance $\boldsymbol{x_q}$, we use $cosim(\boldsymbol{x_q},\boldsymbol{x_i}) = 1 - cos(\boldsymbol{x_q},\boldsymbol{x_i})$ as the baseline distance function for LWLR.

The log-likelihood of data in LWLR is computed with respect to the query instance $\boldsymbol{x_q}$ as

$$l_w(\boldsymbol{\theta}) = \sum_{i=1}^{N} w(\boldsymbol{x_q}, \boldsymbol{x_i}) \log(P_{\boldsymbol{\theta}}(y_i \mid \boldsymbol{x_i}))$$

$$\text{where } w(\boldsymbol{x_q}, \boldsymbol{x_i}) = \exp\left(-\frac{f(\boldsymbol{x_q}, \boldsymbol{x_i})^2}{k^2}\right)$$

The weight $w(\boldsymbol{x_q}, \boldsymbol{x_i})$ is a kernel function which decays with the distance $f(\boldsymbol{x_q}, \boldsymbol{x_i})$. The parameter k is the kernel width, which smoothes out more noise as the value of k increases.

Maximizing $l_w(\boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$ cannot be done in closed form. In our experiments, we solve it using L-BFGS [14] for which we need to compute the partial derivative of $l_w(\boldsymbol{\theta})$ w.r.t $\boldsymbol{\beta}$ parameters. The partial derivative below computes the gradient for the log-likelihood. In the formula below, the expression $[y_i = c_j]$ takes the value of 1 if the expression in the brackets is true, and 0 otherwise.

$$\frac{\partial}{\partial \beta_j^k} l_w(\theta) =$$

$$\sum_{i=1}^{N} w(x_q, x_i)\left(\boldsymbol{x_i^k}[y_i = c_j] - \frac{\boldsymbol{x_i^k}[y_i = c_j]}{Z(\boldsymbol{x_i})} \exp(\beta_j^0 + \sum_{d=1}^{D} \beta_j^d \boldsymbol{x_i^d})\right)$$

$$\text{where } Z(x_i) = \sum_{j=1}^{M} \exp(\beta_j^0 + \sum_{d=1}^{D} \beta_j^d \boldsymbol{x_i^d})$$

3.2 Adding Feature Labeling to Locally Weighted Logistic Regression (LWLR-FL)

Our approach, termed LWLR-FL, incorporates feature labeling into LWLR by leveraging its ability to assign differ-

ent weights to training instances. Intuitively, we use feature labels provided by the end user to define the local neighborhood surrounding the query point. Training instances that are more similar to the query point according to the feature label information are considered to be closer and hence assigned higher weight. We modify the baseline *cosim*($\boldsymbol{x_q}$, $\boldsymbol{x_i}$) distance function to incorporate feature labels. Our modified distance function between $\boldsymbol{x_q}$ and $\boldsymbol{x_i}$ has two distinct components – one based only on their features (satisfied by the baseline distance *cosim*($\boldsymbol{x_q}$, $\boldsymbol{x_i}$)), and the other based on class labels. Since $\boldsymbol{x_q}$ does not have an associated class label, we use the class label of $\boldsymbol{x_i}$ and the feature label information for computing the label similarity.

The label similarity between $\boldsymbol{x_q}$ and $\boldsymbol{x_i}$ is based on the difference between the *positive* and *negative* feature contributions. A *positive feature* is a feature that is labeled with the class label y_i of instance $\boldsymbol{x_i}$ as specified by the feature labels. The positive feature contribution is the sum of the values of all positive features in $\boldsymbol{x_q}$, where $\boldsymbol{x_q}$ is represented as an L2-normalized TFIDF vector. Similarly, a *negative feature* is a feature that is labeled with a class label *other than* y_i. The negative feature contribution refers to the sum of values of all negative features in $\boldsymbol{x_q}$.

We now define the user feature label matrix $\boldsymbol{R}$ as:

$$R_{[DxM]} = \begin{bmatrix} r_1(x^1) & \cdots & r_M(x^1) \\ \vdots & \ddots & \vdots \\ r_1(x^D) & \cdots & r_M(x^D) \end{bmatrix}$$

where $r_j(x^d) = 1$ if the dth feature is labeled to be important for class label y_i; 0 otherwise. Next, let $y_i = c_j$ be the jth class and let the *(D × 1)* vector $\boldsymbol{R(y_i)}$ be the jth column of $\boldsymbol{R}$, which corresponds to the feature labels which are associated with class label y_i.

Let *Ind(**x**)* be an indicator function that is applied to vectors. The ith entry of *Ind(**x**)* has a 1 if the ith entry of $\boldsymbol{x}$ is greater than 0; otherwise the ith entry of the resulting vector has a 0. Let **1** be a vector of size *(M ×1)* having the value 1 in all rows. We define $\boldsymbol{U}$ = *Ind(**R 1**)* which is a *(D × 1)* vector in which the ith row has a 1 if any feature labels involved the ith feature.

On the basis of the above definitions, the difference between the positive and negative feature weights is computed as:

$$\boldsymbol{R}(y_i)^T \boldsymbol{x_q} - \left(\frac{(\boldsymbol{U} - \boldsymbol{R}(y_i))^T \boldsymbol{x_q}}{M-1} \right)$$

In the equation above, the term $\boldsymbol{R(y_i)^T x_q}$ corresponds to the positive feature contribution i.e., the sum of feature values of $\boldsymbol{x_q}$ for those features which are associated with label y_i. The higher this value, the more similar $\boldsymbol{x_q}$ will be to $\boldsymbol{x_i}$ according to the labeled features. The term $(\boldsymbol{U} - \boldsymbol{R(y_i)})^T \boldsymbol{x_q}$ corresponds to the negative feature contribution ie. the sum of the feature values for $\boldsymbol{x_q}$ for the features that are not associated with label y_i. The higher this value, the more dissimilar $\boldsymbol{x_q}$ will be to $\boldsymbol{x_i}$ according to the labeled features. Since we have *(M-1)* class labels excluding y_i, we divide the negative feature contribution by $(M-1)$ to appropriately balance the difference. Putting the two terms together combines the effect of both the positive and the negative feature contributions.

The distance function needs to have smaller values for similar instances. Hence, we define the label similarity component of the distance function as:

$$1\text{-}\boldsymbol{R}(y_i)^T \boldsymbol{x_q} + \left(\frac{(\boldsymbol{U} - \boldsymbol{R}(y_i))^T \boldsymbol{x_q}}{M-1} \right)$$

The complete distance function now becomes:

$$f(\boldsymbol{x_q}, \boldsymbol{x_i}) = \cos im(\boldsymbol{x_q}, \boldsymbol{x_i}) \left[1\text{-}\boldsymbol{R}(y_i)^T \boldsymbol{x_q} + \left(\frac{(\boldsymbol{U} - \boldsymbol{R}(y_i))^T \boldsymbol{x_q}}{M-1} \right) \right]$$

The above function could turn out negative in some cases. Hence, we introduce a max term in the weight computation to handle this scenario.

$$w(\boldsymbol{x_q}, \boldsymbol{x_i}) = \exp\left(- \frac{\max(0, f(\boldsymbol{x_q}, \boldsymbol{x_i}))^2}{k^2} \right)$$

Putting these pieces together, we now have a distance function that incorporates the feature labels into LWLR.

4. EXPERIMENT AND ANALYSIS METHODOLOGIES

To evaluate the LWLR-FL algorithm, we applied it to three real-world text data sets, with two kinds of studies. First, to avoid the prohibitive expense of performing a separate user study on each data set, we followed the usual machine learning convention [15, 16, 18], and simulated end-user feature labeling on multiple data sets using a feature oracle. Second, we then performed a study with real users on one particular data set to investigate the effectiveness of feature labels from end users.

4.1 Oracle Study

In our oracle-based experiments, we used three common text classification datasets: 20 Newsgroups [9], the Modapte split of the Reuters dataset [10], and the Reuters Corpus Volume 1 (RCV1) dataset [11]. From the 20 Newsgroups dataset, we used the four newsgroups: *comp.sys.ibm.pc.hardware*, *misc.forsale*, *sci.med*, and *sci.space*. Since we would also present articles from these newsgroups to end users in our user study (Section 4.2), we wanted to minimize the effects of concept drift by choosing articles that fell within a relatively short date range that included a large number of articles from these newsgroups. As a result, we chose 2925 articles from these four newsgroups within the date range April 1, 1993-April 23, 1993.

From the Modapte split of the Reuters corpus, we used 1092 documents overall from the *earn*, *acq*, *negative_topic*, and *money-fx* classes. From the RCV1 dataset, we used 6500 documents from the *C15*, *CCAT*, *ECAT*, *GCAT*, and *MCAT* classes. The text documents were converted into TF-IDF representation then L2-normalized. We used a vocabulary consisting of unigrams with stopwords removed.

We compared LWLR-FL against three SVM-based algorithms from [16] which are competitive feature labeling methods. Specifically, these SVM-based algorithms are Method 1, Method 2 and a combination of both Methods 1 and 2. We abbreviate these variants as SVM-M1, SVM-M2, and SVM-M1M2 respectively. For these SVM-based methods, we tried linear, RBF and polynomial kernels and found the linear kernel to give the best accuracy. As a result, we only report SVM results with linear kernels.

Since we were interested in the benefits due solely to feature labeling, we did not compare against methods such as Tandem Learning [16] and dual supervision [1, 18] which label *both* features and data instances. For the same reason, we did not compare against any semi-supervised feature labeling methods (e.g., Method 3 from [16] or the GE-based methods in [5]) because such algorithms leverage information from a pool of unlabeled data in addition to the information in feature labels.

Each dataset was split into a training, validation and testing set. Past work [16] has shown that feature labeling is most effective when the training set sizes are small; we created training sets consisting of six instances per class. In our experiments, which deal with a multiclass classification problem with four or five classes (rather than binary classification as in [16]), the total training set sizes were 24 for 20 Newsgroups, 24 for the Modapte split and 30 for RCV1. The training set consisted of an equal number of data instances from each class in order to avoid biases due to class imbalance. The validation set, which was used to tune algorithm parameters, was composed of 100 data points equally distributed among all the classes. The testing set consisted of the remaining data instances. For all datasets, we created 30 different randomly splits for training, validation and testing. The results were averaged over these 30 splits.

To simulate end users for each dataset, the feature oracle selected the ten most predictive features for each class by computing a feature's information gain over the entire corpus, and then assigning its class label based on the most frequent class in which it appeared. This resulted in 40, 40, and 50 oracle feature labels for 20 Newsgroups, Modapte and RCV1 respectively. We experimented with adding one oracle feature label per class, two oracle feature labels per class, and so on until a total of ten per class were added. These oracle feature labels were added in the order of highest information gain. Therefore the oracle study provides an optimistic estimate on the potential gains of using these feature labeling algorithms by providing enough ideal feature labels to benefit the algorithm and by carefully tuning the parameters of the feature labeling algorithms over a large validation set.

4.2 User Study

The strength of oracle studies is the ability to evaluate a variety of data sets, but their weakness is that they may not be realistic as to the choices real users might make. Therefore, for our second experiment, we conducted a user study to harvest feature labels from actual end users on the same 20 Newsgroups classes as used in Section 4.1. We then used the end users' data to compare the performance of the same algorithms as in our oracle study, but with smaller validation sets of size 24 (six instances for each class) to simulate a realistic scenario in which end users are able to label only a limited amount of training instances for both a training and a validation set.

A starting point for our experiment's design was the user study by Raghavan et al. [15]. However, an important difference was that we chose to remove constraints on features end users were allowed to pick. Specifically, rather than having end users select features from a pre-computed list, we allowed them to identify features by freely highlighting text directly in the documents. This gave our participants complete freedom to choose *any* features that they thought were predictive. Consequently, not only were these end users allowed to select existing features in the algorithm's representation, but also to create and label new features, such as through combinations of words or punctuation.

4.2.1 Participants and Procedures

Our user study had 43 participants: 24 males and 19 females. Of these participants, 39 were students currently pursuing an undergraduate or masters degree in a variety of majors. Computer science students and people with background in machine learning or human computer interaction were not allowed to participate.

Participants attended the study in parallel, with up to five participants to a session. At the start of a session, we familiarized the participants with the application to create feature labels, described in Section 4.2.2, through a brief, hands-on tutorial and self-directed exploration. All participants used the same document set during the tutorial, which was different from the main task data set (the training set). After the session they filled out a post session questionnaire asking their thoughts and suggestions for the interface, for our later use in follow-up research.

For the main experiment, the application displayed 24 previously labeled documents in four topics: Computers, Things For Sale, Medicine, and Outer Space (corresponding to the four newsgroups *comp.sys.ibm.pc.hardware*, *misc.forsale*, *sci.med*, and *sci.space,* respectively*)*. Each of the four topics had six documents assigned to it, which were randomly selected from a pool of 200 training instances. The order of the documents was randomized for each participant. Participants were asked to teach the machine "suggestions" by identifying features that they be-

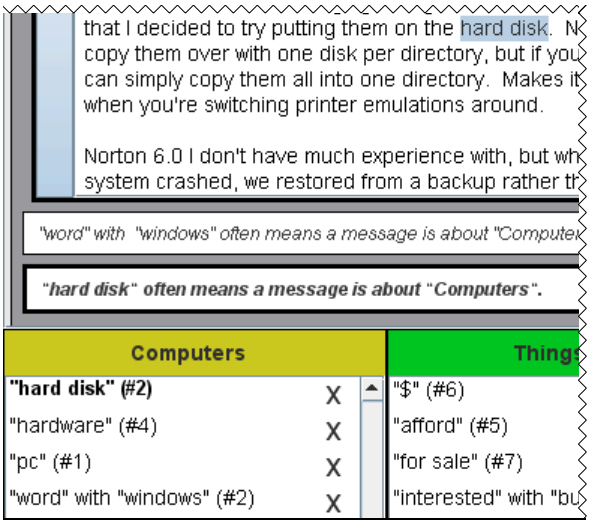


Figure 3: (Top): The document display allows highlighting any text, e.g. "hard disk" in a message labeled Computers (message label cropped off for space, would be to right of text). (Bottom): The feature display shows suggestions enumerated below the message. The currently selected suggestion is bolded in the feature panel at the bottom of the screen.

lieved would help it label future documents. Within a time limit of twelve minutes, participants were asked to provide at least two suggestions per topic, with an emphasis placed on selecting the best features for each newsgroup.

4.2.2 Environment

For the study, we created a software prototype allowing participants to flexibly provide feature labels within a message reader interface. The prototype window had two main areas (Figure 3): the document display and the feature display. The document display area, which was the participants' main interface for labeling features, showed the list of documents, each with its newsgroup label, in a scrollable panel. Participants could highlight portions of the document text with the cursor (as in Figure 3) that they thought were characteristic of the document's newsgroup. Participants could create multiple suggestions for each document, and could also delete or modify their suggestions. Participants were given great flexibility in identifying features: they were allowed to highlight *anything* in the document text, including single words, punctuation, continuous phrases, and non-contiguous words or phrases.

The feature display at the bottom was a quick reference, to remind participants of features they had already identified, along with a clickable link to the context in which they highlighted it. Participants' selections of non-continuous words or phrases were shown as blocks separated by "with".

4.2.3 Algorithm Evaluation

We used participant-provided feature labels to compare the performance of LWLR-FL, SVM-M1, SVM-M2 and SVM-M1M2. Participants could label features by highlighting *any* text—they did not have to know whether their feature existed before (recall that we used a vocabulary of unigrams with stopwords removed for the original representation). If a participant created a new feature, we added it to the document representation used for that participant's data and created a corresponding feature label for it. Using these data, we analyzed two variants of each algorithm: one variant used participants' labels on existing features only, and the other used all features that participants provided.

4.2.4 Feature Characteristics Analysis

In addition to information gain, we computed relatedness as a measure for the features that participants provided. Informally, relatedness of a feature to a topic is how closely it represents a topic's subject matter.

We used ConceptNet to provide us with a measure of relatedness. ConceptNet [12] is a commonsense knowledgebase, generated automatically from sentences entered by users of the Open Mind Common Sense Project. ConceptNet can support textual reasoning such as topic-jisting and analogy-making by providing relationships between words and phrases. AnalogySpace [19], which is based on ConceptNet, provides a similarity score (-1 to 1) between two features e.g. "horse" and "cow" are similar to a degree of 0.89 yet "pencil" and "cow" are not very related with a similarity score of -0.01. We used this similarity score as a measure of relatedness between features and topics.

To obtain similarity scores, we used the following process. We excluded punctuation or symbols e.g. "$", "?", etc, as ConceptNet does not contain information on punctuation. We then normalized the features by using the in-built ConceptNet function which takes a string and converts it into its most "natural" state, removing modifiers, inflections, and stop words. For example "asking", when normalized, is "ask". If the function call produced no normalized output, it was entered into ConceptNet in its unmodified form. For non-continuous words, we calculated a similarity score for each individual word, then we used the maximum score of all words in the feature. When participants provided compound words (e.g. "diet/exercise") and phrases (e.g. "hard disk"), we calculated a similarity score for each individual word and the original given feature, again using the maximum as the final score.

5. RESULTS AND DISCUSSION

In this section, we present results on the effectiveness of the LWLR-FL algorithm, first for simulated, ideal circumstances with features provided by a feature oracle, and, second, when feature labels were provided by participants. In order to improve interaction with systems using this approach in real settings, we also investigated the characteristics of end-user features and the algorithm's sensitivity to parameter settings.

5.1 Oracle Feature Labels

Figure 4 presents the effects of incrementally adding the top ten oracle feature labels per class over a variety of algo-

rithms and data sets. We evaluate the algorithms in terms of the average macro-average F1 score (abbreviated to macro-F1), where the average is computed over the 30 random training/validation/testing splits. As more oracle feature labels were added, the average macro-F1 scores generally increased for all algorithms. To avoid clutter on the graphs, we only show the top two algorithms for each dataset.

In order to evaluate the effectiveness of feature labels, we compared against two baseline algorithms that do not take feature labels into account (Figure 4, dashed and solid line without markers). For comparison with LWLR-FL, we used a LWLR that uses cosine similarity as a distance metric. Likewise, a "plain" SVM can be considered as a baseline algorithm for the SVM-based algorithms. The benefit of incorporating feature labeling by using LWLR-FL and the SVM-based algorithms can be expressed as the improvement in macro-F1 score over their respective baselines; we denote this improvement as $\Delta_{baseline}$. The average $\Delta_{baseline}$ over the 30 runs was significant for LWLR-FL in all cases, and significant for the best SVM-based methods except for one and two feature labels per class on the Modapte dataset (Wilcoxon signed-rank test, $p < 0.05$). The highest $\Delta_{baseline}$ values were achieved by LWLR-FL on the Modapte data with ten features per class, with a mean $\Delta_{baseline}$ of 0.27 and a max of 0.38. Thus, incorporating feature labels in general is better than not taking feature labels into account. LWLR-FL produced larger average $\Delta_{baseline}$ scores than SVM-M1M2 on the 20 Newsgroups and Modapte datasets, while LWLR-FL and SVM-M2 had similar average $\Delta_{baseline}$ scores on the RCV1 dataset. Interestingly, on the Modapte dataset, LWLR performed worse than SVM but once oracle feature labels were provided, LWLR-FL was able to use the feature labels more effectively than any of the SVM-based methods and outperformed all the algorithms once more than five feature labels were added.

Overall, LWLR-FL outperforms other feature labeling algorithms on 20 Newsgroups and on Modapte when above five features per class are added. LWLR-FL performed similar to SVM-M2 on RCV1. Table 1 summarizes the macro-F1 scores for the different algorithms over the three datasets when ten oracle feature labels per class were added. In Table 1, LWLR-FL produced the highest mean macro-F1 score or matched it on all three datasets; its effectiveness was significantly better than SVM-M1M2 on the 20 Newsgroups dataset (Wilcoxon signed-rank test, $p < 0.05$).

Macro F1, Adding 10 Oracle Features per class			
Algorithm	20 Newsgroups	Modapte	RCV1
SVM	0.635	0.608	0.559
LWLR	0.652	0.545	0.554
SVM-M1	0.667*	0.649*	0.573*
SVM-M2	0.745*	0.802*	0.660*
SVM-M1M2	0.749*	0.797*	0.656*
LWLR-FL	**0.777***	0.824*	0.660*

Table 1: Results of adding 10 oracle features per class for all three datasets. * denotes values that are significantly greater than baseline algorithms, bolded values are significantly greater than all other algorithms at the 0.05 level (Wilcoxon signed-rank test, $p < 0.05$).

5.2 End User Features

The results in Section 5.1 indicate that with ideal feature labels, incorporating feature label information can improve the classifier, especially when the LWLR-FL algorithm is used. We now turn our attention to the effects of feature labels provided by actual end users, which we expected to have less of a gain than the idealized oracle feature labels. Figure 5 illustrates the performance of the feature labeling algorithms. (In our analysis, SVM-M1M2 consistently outperformed SVM-M1 and SVM-M2. Therefore, to avoid clutter in the graphs, we only show results for SVM-M1M2.) For reference, the leftmost group duplicates the eight oracle feature labels per class results from Section 5.1. We chose eight oracle feature labels per class as a reference because on average, participants provided this many feature labels per class. The middle group presents results when only feature labels on existing features were considered (feature labels on features created by participants were ignored). Finally, the rightmost group of results illustrates the macro-F1 scores when all feature labels are considered, including the new features created by the participant.

As we expected, these gains did not match those of the oracle feature labels. This points out the importance of including evaluations with real end users for this type of problem. Evaluations in idealized conditions produce results about an algorithm's potential, and hence are overly optimistic, as

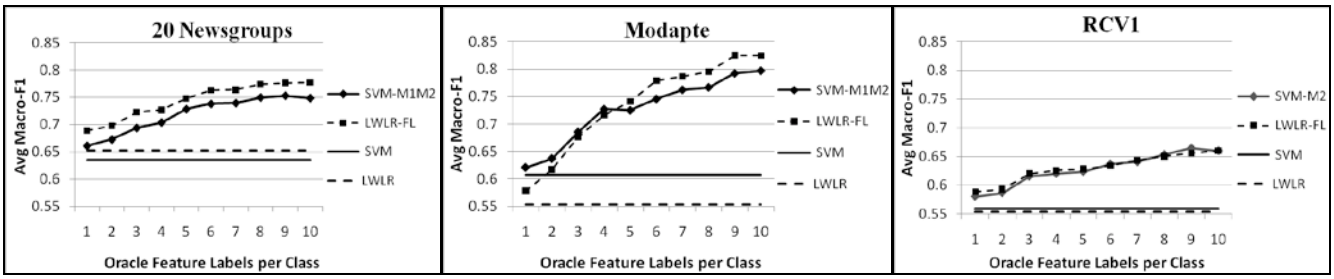


Figure 4: Average Macro-F1 after adding oracle feature labels on (left) 20 Newsgroups, (middle) Modapte, and (right) RCV1.

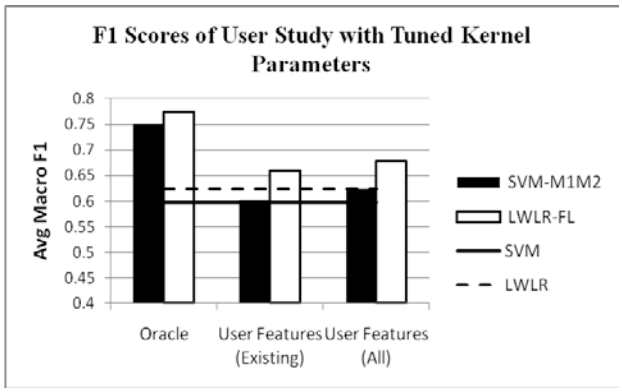


Figure 5: Average macro-F1 scores for incorporating end user feature labels to the 20 Newsgroups dataset: (Left) incorporating 8 oracle feature labels per class from Section 5.1, (Middle) incorporating end-user feature labels only for existing features, (Right) incorporating all end-user feature labels.

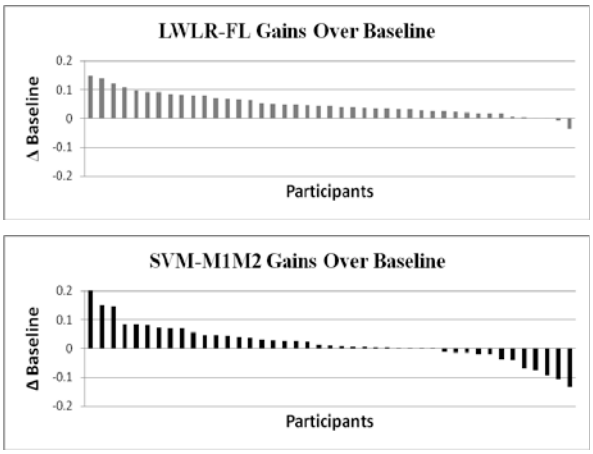


Figure 6: $\Delta_{baseline}$ scores sorted in decreasing order for LWLR-FL (Top) and SVM-M1M2 (Bottom). Note that the order of the participants is not the same in the graphs.

seen by comparing Figure 5's leftmost bars with the bars to the right.

Participants were able to provide useful feature labels in our experiments, as can be seen in Figure 5. Both algorithms outperformed their baselines by a statistically significant margin (Wilcoxon signed-rank test, $p < 0.05$) for the "all" features case, but only LWLR-FL was significantly better with "existing" features (Wilcoxon signed-rank test, $p < 0.05$)[1]. LWLR-FL was significantly better than SVM.M1M2 (Wilcoxon signed-rank test, $p < 0.05$) for both existing and all features. Feature labels were useful when labels were incorporated for existing features only (Figure 5 middle) and also when labels were incorporated for all features, including those created by end users (Figure 5 right). In fact, there is a slight increase in average macro-F1 when features created by end users are added, indicating that end users can indeed create predictive features. This result is encouraging for the deployment of feature labeling algorithms with ordinary end users and for future work in end-user feature engineering.

LWLR-FL was also more robust to lower quality feature labels supplied by end users than SVM.M1M2. Figure 6 plots the values of $\Delta_{baseline}$ in decreasing order for all 43 participants; note that the order of participants is not the same on the graphs. The $\Delta_{baseline}$ values for LWLR-FL are shown at the top while those for SVM-M1M2 are at the bottom. These graphs show that more participants had positive $\Delta_{baseline}$ scores with LWLR-FL than with SVM-M1M2. In fact, SVM-M1M2 produced larger negative $\Delta_{baseline}$ scores than LWLR-FL.

[1]We also ran participant data using 100 instances for validation. Here, both algorithms performed significantly better than their baselines with LWLR-FL still significantly outperforming SVM-M1M2.

5.2.1 Characteristics of End User Features

Our results show that features provided by end users can improve the accuracy of feature labeling algorithms. To understand what kinds of features an algorithm should expect from end users, we investigated the types of features our participants provided, the amount of gain each type contributed, and a possible basis on which participants may have chosen these features.

Table 2 shows the frequencies of the types of features participants chose. The most common type, accounting for about 60% of the features, was features that the algorithm already knew existed, in the form of unigrams (row 1 in the table). Some of these had information gains comparable to features chosen by the oracle (for example, 10 of the 43 participants chose the top oracle feature "sale"), although overall, participants' features had a somewhat lower average information gain (0.035) than the oracle's (0.078).

However, quite often participants' choices were different from the oracle's, across all feature types. For example, the top oracle choice for the Medicine topic (high information gain) was "writes". This feature was *never* chosen by a participant. To understand the difference between participants' feature choices and the oracle's, we turned to ConceptNet, computing relatedness as described in Section 4.2.4.

These computations showed that participants chose features with higher relatedness (average 0.308) than those chosen by the oracle (average 0.231). In general, the participants' choices of features with high relatedness *helped* as relatedness has a relationship with information gain (linear regression, $R^2=0.04$, $p<0.001$). This suggests possible directions for IUI designers to use in encouraging end users to usefully label features. For example, relatedness could be used to suggest relevant predictive features for end users to label, thereby helping them overcome their known difficulty of knowing where to look when trying to provide guidance to

the system (e.g., "What kind of words should we tell the computer [relating] to Systems?" [7]).

Finally, 40% of participants' feature choices were *not* known to the algorithm previously, as rows 2-5 in the table show. Some of these features, such as stopwords and punctuation, had previously been removed from the vocabulary but were partially reintroduced by participants. Features such as multiple word phrases (n-grams) and non-continuous words (feature combinations) cannot be addressed by simply adding all possibilities (e.g., all n-grams) as features to the learning algorithm, because doing so would explode the feature representation, making learning infeasible. Also, for applications that customize themselves to the end user, these specific features may be unique to those end users and thus not foreseeable by the algorithm designer prior to deployment. Thus, allowing end users to provide features not originally in the learning algorithm's data representation is an important benefit, which we have only begun to investigate in end-user feature labeling.

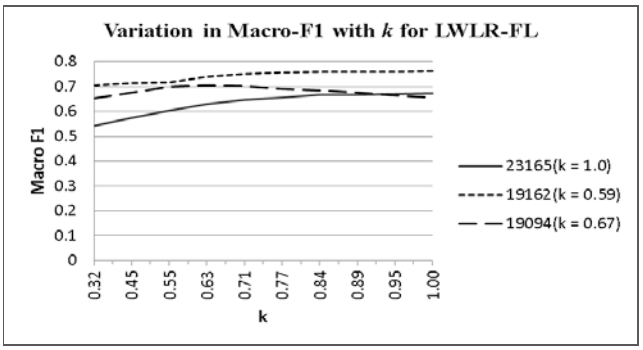
5.2.2 Sensitivity Analysis

Most of the algorithms that incorporate feature labels are sensitive to key parameters that control the influence of the feature labels, but these parameters are difficult to set prior to deployment due to the uniqueness of each end user's data distribution. Therefore, some algorithms that performed well in idealized situations may perform poorly in real-world circumstances. Although the user could label more data for a more representative validation set, this could require an unrealistic time investment by the user. Ideally then, the algorithm's performance should not be overly sensitive to the values of these parameters. We performed a sensitivity analysis to investigate the robustness of LWLR-FL and the SVM-based methods to their parameter settings.

The sensitivity of LWLR-FL depends only on the kernel width k, which defines the neighborhood around a query point. The best k^2 for a training set was found using grid search in the range 0.1 - 1.0 (values larger than 1.0 generally reduced macro-F1). Figure 7 (top) illustrates the variation in macro-F1 score for LWLR-FL for three participants.

Feature Types (Examples from Participants' Data)	Mean number per participant	Mean information gain	Mean ConceptNet similarity
Existing feature ("sale")	19.419	0.040	0.280
Reintroduced stopword ("asking")	0.140	0.036	0.412
Continuous phrase ("space shuttle")	5.977	0.048	0.394
Non-continuous words ("cold" with "flu")	4.116	0.011	0.359
Features involving punctuation ("for sale" with "$")	2.651	0.056	0.243
Means overall	32.302	0.035	0.308

Table 2: Labeled features per participant by type. All types are disjoint. Only 6 of the 43 users re-introduced removed stopwords, and each re-introduced only one.



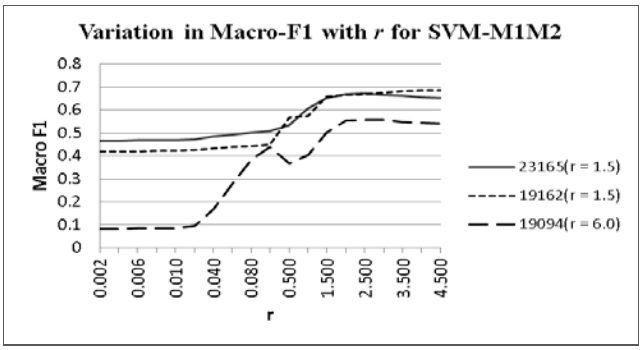


Figure 7: Variation of macro-F1 with k for LWLR-FL (Top) and with r for SVM-M1M2 (Bottom). Data is plotted for the same three participants. The parameter value chosen after tuning on a validation set of size 24 (as described in Section 4.2) is shown for each participant in brackets in the plot

The participants were chosen to represent extreme conditions, one having highest, one having average, and one having lowest gains in macro-F1. The macro-F1 score is computed on each participant's holdout test set.

As described in Section 2, SVM-M1M2 needs to tune parameters a, d, and r. All these parameters were tuned on the validation set using a multi-dimensional grid search. Figure 7 (bottom) illustrates the variation of macro-F1 scores on the same three participants when we vary the r parameter for the SVM-M1M2 method, which was the most sensitive parameter. In this graph, we hold the a and d parameters fixed to their values tuned on the validation set.

Although the scales of the parameters are difficult to compare head-to-head, Figures 7 (top) and (bottom) show that SVM-M1M2 is more sensitive to the r parameter than LWLR-FL is with the k parameter. Even within a small radius around the tuned value, the variation in macro-F1 for SVM-M1M2 covers a larger range. For LWLR-FL, the variation of macro-F1 scores is smooth with varying k and tends to be within a narrower range. In addition, LWLR-FL also has fewer parameters to tune than SVM-M1M2. (We found empirically that a "default" value of $k = \sqrt{0.5}$

yielded reasonably good macro-F1 scores.) These results suggest that LWLR-FL is robust to changes in the *k* parameter and suitable for real-world deployments in which a large number of training instances may not be available.

6. CONCLUSION

Our paper has shown the viability of feature labeling in real circumstances, with end users freely choosing features to label directly from text documents. Our new LWLR-FL algorithm expands LWLR to take feature labeling into account. Our results show that LWLR-FL outperformed or matched SVM-based methods under ideal conditions in an oracle study. In our user study, we allowed ordinary end users to select any features for labeling directly from text documents. LWLR-FL significantly outperformed the SVM-based methods in this more realistic setting. Further, our results showed that LWLR-FL is less sensitive to its parameter settings than SVM-M1M2.

As to the end-user labels themselves, we showed that real end users' feature labels helped on average for both LWLR-FL and SVM-M1M2, with the features end users chose for labeling to be conceptually related to the class labels, although with moderately lower information gains compared to those of the oracle's. These results are promising, as they show that end users who know nothing about machine learning can use feature labeling to significantly improve machine learning algorithms trained on small data sets.

Finally, our results point to promising future research: first, extending the LWLR-FL algorithm to a semi-supervised setting and second, designing suitable user interfaces to help end users choose and create features to label.

Taken together, these results suggest that flexible feature labeling by end users is an effective solution for allowing end users to improve the learning algorithm behind their intelligent user interface.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant 0803487.

REFERENCES

1. Attenberg, J., Melville, P., and Provost, F. A unified approach to active dual supervision for labeling features and examples, in *Proc. European Conf. Machine Learning* (2010).
2. Cleveland, W., and Devlin, S. Locally-weighted regression: An approach to regression analysis by local fitting. *J. American Statistical Assn*. *83*, 403 (1988), 596–610.
3. Cohn D. A., Ghahramani, Z., and Jordan, M. I. Active learning with statistical models. *J. Artificial Intelligence Research*, 4 (1996), 129-145.
4. Deng, K. Omega: On-line Memory-Based General Purpose System Classifier (PhD Dissertation). Carnegie Mellon University, Pittsburgh, PA, 1998.
5. Druck, G., Mann, G., and McCallum, A. Learning from labeled features using generalized expectation criteria, in *Proc. SIGIR* (2008), ACM, 595-602.
6. Hastie, T., Tibshirani, R., and Friedman, J. H. *The Elements of Statistical Learning*. Springer, 2003.
7. Kulesza, T., Wong, W.-K., Stumpf, S., Perona, S., White, S., Burnett, M., Oberst, I. Ko, A. Fixing the program my computer learned: Barriers for end users, challenges for the machine, in *Proc. IUI* (2009), ACM, 187-196.
8. Kulesza, T., Stumpf, S., Burnett, M., Wong, W.-K., Riche, Y., Moore, T., Oberst, I., Shinsel, A., McIntosh, K., Explanatory debugging: supporting end-user debugging of machine-learned programs, in *Proc. IEEE Symp. Visual Languages and Human-Centric Computing* (2010), IEEE, 41-48.
9. Lang, K. Newsweeder: Learning to filter netnews, in *Proc. ICML* (1995), 331-339.
10. Lewis, D. Reuters-21578. Available at http://www.daviddlewis.com/resoursce/testcollections/reuters21578.
11. Lewis, D. D., Yang, Y., Rose, T., Li, F. RCV1: A new benchmark collection for text categorization research. *JMLR,* 5 (2004), 361-397. http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf.
12. Liu, H., and Singh, P. ConceptNet—a practical commonsense reasoning tool-kit. *BT Technology Journal 22*, 4 (2004), 211-226.
13. McCallum, A., Mann, G., and Druck, G. Generalized Expectation Criteria (Technical Report UM-CS-2007-60). University of Massachusetts, Amherst, 2007.
14. Nocedal, J. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation*, 35 (1980), 773-782.
15. Raghavan, H., Madani, O., and Jones, R. Active Learning with Feedback on Both Features and Instances. *JMLR* 7 (2006), 1655-1686.
16. Raghavan, H. and Allan, J. An interactive algorithm for asking and incorporating feature feedback into support vector machines, in *Proc. SIGIR* (2007), ACM, 79-86.
17. Roth, D. and Small, K. Interactive feature space construction using semantic information, in *Proc. CoNLL* (2009), 66-74.
18. Sindhwani, V., Melville, P., and Lawrence, R. Uncertainty sampling and transductive experimental design for active dual supervision. *Int. Conf. Machine Learning* (2009), 953-960.
19. Speer, R., Havasi, C., and Lieberman, H. AnalogySpace: Reducing the dimensionality of common sense knowledge, in *Proc. AAAI* (2008).
20. Stumpf, S. Rajaram, V., Li, L., Wong, W.-K., Burnett, M., Dietterich, T., Sullivan, E., and Herlocker J. Interacting meaningfully with machine learning systems: Three experiments. *Int. J. Human-Computer Studies 67*, 8 (2009), 639-662.