

Closing the Loop: Fast, Interactive Semi-Supervised Annotation With Queries on Features and Instances

Burr Settles

Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213 USA
bsettles@cs.cmu.edu

Abstract

This paper describes DUALIST, an open-source active learning annotation tool which solicits and learns from labels on both features (e.g., words) and instances (e.g., documents). We present a novel semi-supervised training algorithm developed for this application, which is (1) fast enough to support real-time interactive speed, and (2) at least as accurate as pre-existing methods for learning with mixed feature and instance queries. Human annotators in user studies were able to produce near-state-of-the-art classifiers—on several corpora in a variety of application domains—with only a few minutes of effort.

1 Introduction

In active learning, a classifier may participate in its own training process by posing *queries*, such as requesting labels for documents in a text classification task. The goal is to maximize the accuracy of the trained system in the most economically efficient way. This paradigm is well-motivated for natural language applications, where unlabeled data may be readily available (e.g., text on the Internet), but the annotation process can be slow and expensive.

Nearly all previous work in active learning, however, has focused on selecting queries from the *learner's* perspective. For example, experiments are often run in simulation rather than with user studies, and results are routinely evaluated in terms of training set size rather than human annotation time or labor costs (which are more reasonable measures of labeling effort). Many state-of-the-art algorithms

are also too slow to run or too tedious to implement to be useful for real-time interaction with human annotators, and few analyses have taken these factors into account. Furthermore, there is very little work on actively soliciting domain knowledge from humans (e.g., information about features) and incorporating this into the learning process.

While selecting good queries is clearly important, if our goal is to reduce actual annotation effort we must take these human factors into account. In this work, we propose a new interactive annotation interface which addresses some of these issues; in particular it has the ability to pose queries on both features (e.g., words) and instances (e.g., documents). We present a novel semi-supervised learning algorithm that is fast, flexible, and accurate enough to support these interface design constraints interactively.

2 DUALIST: Utility for Active Learning with Instances and Semantic Terms

Figure 1 shows a screenshot of the DUALIST interface. On the left panel, users are presented with unlabeled documents: in this case Usenet messages that belong to one of two sports-related topics: *baseball* and *hockey*. Users may label documents by clicking on the class buttons listed below each text. In cases of extreme ambiguity, users may ignore a document by clicking the “X” to remove it from the pool of possible queries.

On the right panel, users are given a list of word queries organized into columns by class label. The rationale for these columns is that they should reduce cognitive load (i.e., once a user is in the *baseball* mindset, s/he can simply go down the list, label-



Figure 1: A screenshot of the DUALIST interface.

ing features in context: “plate,” “pitcher,” “bases,” etc.). Within each column, words are sorted by how informative they are to the classifier, and users may click on words to label them. Each column also contains a text box, where users may “inject” domain knowledge by typing in arbitrary words (whether they appear in any of the columns or not). The list of previously labeled words appears at the bottom of each list (highlighted), and can be unlabeled at any time, if the user later feels they made an error.

Finally, there is a large submit button at the top of the screen, which users must click to re-train the classifier and receive a new set of queries. The learning algorithm is actually fast enough to do this automatically after each labeling action. However, we found such a dynamically changing interface to be annoying for users (e.g., words they wanted to label would move or disappear).

2.1 A Generative Model for Learning from Feature and Instance Labels

For the underlying model in this system, we use multinomial naïve Bayes (MNB) since it is simple, fast, and known to work well for several natural language applications—text classification in particular—despite its simplistic and often violated independence assumptions (McCallum and Nigam, 1998; Rennie et al., 2003).

MNB models the distribution of features as a multinomial: documents are sequences of words, with the “naïve” assumption that words in each position are generated independently. Each document is treated as a mixture of classes, which have

their own multinomial distributions over words. Let the model be parameterized by the vector θ , with $\theta_j = P(y_j)$ denoting the probability of class y_j , and $\theta_{jk} = P(f_k|y_j)$ denoting the probability of generating word f_k given class y_j . Note that for class priors $\sum_j \theta_j = 1$, and for per-class word multinomials $\sum_k \theta_{jk} = 1$. The likelihood of document x being generated by class y_j is given by:

$$P_\theta(x|y_j) = P(|x|) \prod_k (\theta_{jk})^{f_k(x)},$$

where $f_k(x)$ is the frequency count of word f_k in document x . If we assume $P(|x|)$ is distributed independently of class, and since document length $|x|$ is fixed, we can drop the first term for classification purposes. Then, we can use Bayes’ rule to calculate the posterior probability under the model of a label, given the input document for classification:

$$P_\theta(y_j|x) = \frac{P_\theta(y_j)P_\theta(x|y_j)}{P_\theta(x)} = \frac{\theta_j \prod_k (\theta_{jk})^{f_k(x)}}{Z(x)}, \quad (1)$$

where $Z(x)$ is shorthand for a normalization constant, summing over all possible class labels.

The task of training such a classifier involves estimating the parameters in θ , given a labeled set of instances $\mathcal{L} = \{\langle x^{(l)}, y^{(l)} \rangle\}_{l=1}^L$. To do this, we use a Dirichlet prior and take the expectation of each parameter with respect to the posterior, which is a simple way to estimate a multinomial (Heckerman, 1995). In other words, we count the fraction of times the word f_k occurs in the labeled set among documents of class y_j , and the prior adds m_{jk} “hallucinated” occurrences for a smoothed version of the maximum likelihood estimate:

$$\theta_{jk} = \frac{m_{jk} + \sum_i P(y_j|x^{(i)}) f_k(x^{(i)})}{Z(f_k)}. \quad (2)$$

Here, m_{jk} is the prior for word f_k under class y_j , $P(y_j|x^{(i)}) \in \{0, 1\}$ indicates the true labeling of the i th document in the training set, and $Z(f_k)$ is a normalization constant summing over all words in the vocabulary. Typically, a uniform prior is used, such as the Laplacian (a value of 1 for all m_{jk}). Class parameters θ_j are estimated a similar way, by counting the fraction of documents that are labeled with that class, subject to a prior m_j . This prior is important in the event that no documents are yet labeled

with y_j , which is not unusual early on in the active learning process.

Recall that our scenario lets human annotators provide not only document labels, but feature labels as well. To make use of this additional information, we assume that labeling the word f_k with a class y_j increases the probability $P(f_k|y_j)$ of that word appearing in documents of that class. The natural interpretation of this under our model is to increase the prior m_{jk} for the corresponding multinomial. To do this we introduce a new parameter α , and define the elements of the Dirichlet prior as follows:

$$m_{jk} = \begin{cases} 1 + \alpha & \text{if } f_k \text{ is labeled with } y_j, \\ 1 & \text{otherwise.} \end{cases}$$

This approach is extremely flexible, and offers three particular advantages over the previous “pooling multinomials” approach for incorporating feature labels into MNB (Melville et al., 2009). First, pooling multinomials assumes binary classification, but our method can be applied to problems with multiple classes. Second, feature labels need not be mutually exclusive, so the word “score” could be labeled with both *baseball* and *hockey*, if necessary. Third, users can conceivably provide feature-label specific parameters α_{jk} to, for example, imply that the word “inning” is a stronger indicator for *baseball* than the word “score” (which is a more general sports term). However, we leave this aspect for future work and employ the fixed- α approach as described above.

2.2 Exploiting Unlabeled Data

In addition to document and feature labels, we usually have access to a large unlabeled corpus. In fact, these texts form the pool of possible instance queries in active learning. We can take advantage of this additional data in generative models like MNB by employing the Expectation-Maximization (EM) algorithm. Combining EM with pool-based active learning was previously studied in the context of instance labeling (McCallum and Nigam, 1998), and we extend the method to our interactive scenario, which supports feature labeling as well.

First, we estimate initial model parameters θ' as in Section 2.1, but using *only* the priors (and no labeled data). Then, we apply the induced classifier on the unlabeled pool $\mathcal{U} = \{x^{(u)}\}_{u=1}^U$ (Eq. 1). This is the “E” step of the EM algorithm. Next we re-estimate

feature multinomials θ_{jk} (Eq. 2), using both labeled instances from \mathcal{L} and probabilistically-labeled instances from \mathcal{U} . In other words, $P(y_j|x) \in \{0, 1\}$ indicates the true document label for $x \in \mathcal{L}$, and $P(y_j|x) = P_{\theta'}(y_j|x)$ for $x \in \mathcal{U}$. We also weight the data from \mathcal{U} by 1/10, so as not to overwhelm the training signal coming from true instance labels in \mathcal{L} . Class parameters θ_j are re-estimated in the analogous fashion. This is the “M” step.

For speed and interactivity, we actually stop training after this first iteration. When feature labels are available, we found that EM generally converges in 4–10 iterations, requiring more training time but rarely improving accuracy (the largest gains consistently come in the first iteration). Also, we ignore labeled data in the initial estimation of θ' because \mathcal{L} is too small early in active learning to yield good results with EM. Perhaps this can be improved by using an ensemble (McCallum and Nigam, 1998), but that comes at further computational expense. Feature labels, on the other hand, seem generally more reliable for probabilistically labeling \mathcal{U} .

2.3 Selecting Instance and Feature Queries

The final algorithmic component to our system is the selection of informative queries (i.e., unlabeled words and documents) to present to the annotator.

Querying instances is the traditional mode of active learning, and is well-studied in the literature; see Settles (2009) for a review. In this work we use entropy-based uncertainty sampling, which ranks all instances in \mathcal{U} by the posterior class entropy under the model $H_{\theta}(Y|x) = -\sum_j P_{\theta}(y_j|x) \log P_{\theta}(y_j|x)$, and asks the user to label the top D ranked documents. This simple heuristic is an approximation to querying the instance with the maximum information gain (since the class entropy, once labeled, is zero), under the assumption that each x is representative of the underlying natural distribution in \mathcal{U} . Moreover, it is extremely fast to compute, which is important for our interactive environment.

Querying features, though, is a newer idea with significantly less research behind it. Previous work has either assumed that (1) features are not assigned to classes, but instead flagged for “relevance” to the task (Godbole et al., 2004; Raghavan et al., 2006), or (2) feature queries are posed just like instance queries: a word is presented to the annotator, who

must choose among the labels (Druck et al., 2009; Attenberg et al., 2010). Recall from Figure 1 that we want to organize feature queries into columns by class label. This means our active learner must produce queries that are class-specific.

To select these feature queries, we first rank elements in the vocabulary by information gain (IG):

$$IG(f_k) = \sum_{\mathbf{I}_k} \sum_j P(\mathbf{I}_k, y_j) \log \frac{P(\mathbf{I}_k, y_j)}{P(\mathbf{I}_k)P(y_j)},$$

where $\mathbf{I}_k \in \{0, 1\}$ is a variable indicating the presence or absence of a feature. This is essentially the common feature-selection method for identifying the most salient features in text classification (Sebastiani, 2002). However, we use both \mathcal{L} and probabilistically-labeled instances from \mathcal{U} to compute $IG(f_k)$, to better reflect what the model believes it has learned. To organize queries into classes, we take the top V ranked features (ignoring those already labeled) and add f_k to the class y_j with which it occurs most frequently, as well as any other class with which it occurs at least 3/4 as often. Intuitively, this approach (1) queries features that the model believes are most relevant, and (2) automatically identifies the classes for which a feature seems most correlated. To our knowledge, this is the first work to try and achieve both of these properties.

3 Experiments

We conduct four sets of experiments to evaluate our approach. The first two are “offline” experiments, designed to better understand (1) how our training algorithm compares to existing methods for feature-label learning, and (2) the effects of tuning the α parameter. The other experiments are user studies designed to empirically gauge how well human annotators make use of this interface.

We use a variety of benchmark text classification corpora in the following evaluations. WebKB (Craven et al., 1998) consists of 4,199 university web pages of four types: *course*, *faculty*, *project*, and *student*. 20 Newsgroups (Lang, 1995) is a set of 18,828 Usenet messages from 20 different online discussion groups. For certain experiments (such as the one shown in Figure 1), we also use topical subsets. Movie Reviews (Pang et al., 2002) is a set of 2,000 online movie reviews categorized as *positive*

or *negative* in their assessment. All data sets were processed using lowercased unigram features, with punctuation and common stop-words removed.

3.1 Comparison of Feature-Label Training Algorithms

An important question is how well our approach, “MNB/Priors,” performs relative to existing baseline methods for learning with feature labels. We compare against “MaxEnt/GE,” a maximum entropy model trained using generalized expectation criteria (Druck et al., 2008), and “MNB/Pool,” which is naïve Bayes trained using the pooling multinomials approach (Melville et al., 2009).

We use the implementation of GE training from the open-source MALLET toolkit¹, and implement both MNB variants in the same data-processing pipeline. Because the GE implementation only supports labeled features (and not labeled instances as well), we limit the MNB methods to features for a fair comparison. To obtain feature labels in this experiment, we simulate a “feature oracle” as in previous work (Druck et al., 2008; Attenberg et al., 2010), which is essentially the query selection algorithm from Section 2.3, but using complete labeled data to compute $IG(f_k)$. We conservatively use only the top 10 features per class, which is meant to resemble a handful of very salient features that a human might brainstorm to jumpstart the learning process. We experiment with EM₁ (one-step EM) variants of both MNB/Pool and MNB/Priors, and set $\alpha = 50$ for the latter (see Section 3.2 for details on tuning this parameter). Results are averaged over 10 folds using cross-validation, and all experiments are conducted on a single 2.53GHz processor machine.

Table 1 presents these results. As expected, adding one iteration of EM for semi-supervised training improves the accuracy of both MNB methods across all data sets. These improvements come without significant overhead in terms of time: training still routinely finishes in a fraction of a second per fold. MNB/Pool and MNB/Priors, where they can be compared, perform virtually the same as each other with or without EM, in terms of accuracy and speed alike. However, MNB/Pool is only applicable to binary classification problems. As explained in

¹<http://mallet.cs.umass.edu>

Corpus	MaxEnt/GE		MNB/Pool		Pool+EM ₁		MNB/Priors		Priors+EM ₁	
WebKB	22.2	(4.9)	—	—	—	—	67.5	(≤0.1)	67.8	(0.1)
20 Newsgroups	49.7	(326.6)	—	—	—	—	50.1	(0.2)	70.7	(6.9)
Science	86.9	(5.7)	—	—	—	—	71.4	(≤0.1)	92.8	(0.1)
Baseball/Hockey	49.9	(0.8)	90.7	(≤0.1)	96.7	(≤0.1)	90.5	(≤0.1)	96.9	(≤0.1)
Mac/PC	50.5	(0.6)	86.7	(≤0.1)	91.2	(≤0.1)	86.6	(≤0.1)	90.2	(≤0.1)
Movie Reviews	68.8	(1.8)	68.0	(≤0.1)	73.4	(0.1)	67.7	(≤0.1)	72.0	(0.1)

Table 1: Accuracies and training times for different feature-label learning algorithms on benchmark corpora. Classification accuracy is reported for each model, using only the top 10 oracle-ranked features per label (and no labeled instances) for training. The best model for each corpus is highlighted in bold. Training time (in seconds) is shown in parentheses on the right side of each column. All results are averaged across 10 folds using cross-validation.

Section 2.1, the MNB/Priors method we present here is more flexible, and thus preferred for a general-use interactive annotation tool like DUALIST.

The semi-supervised MNB methods are also consistently more accurate than GE training—and are about 40 times faster as well. The gains of Priors+EM₁ over MaxEnt/GE are statistically significant in all cases except Movie Reviews². MNB’s superiority holds true when using 5–20 oracle-ranked features per class, but as the number of feature labels increases beyond 30 features per label, GE is often more accurate (results not shown). If we think of MaxEnt/GE as a discriminative analog of MNB/Priors+EM, this is consistent with what is known about labeled set size in supervised learning for generative/discriminative pairs (Ng and Jordan, 2002). However, the time complexity of GE training increases with each new labeled feature, since it adds a new constraint to the objective function whose gradient must be computed across all the unlabeled data. In short, GE is too slow and too inaccurate (with fewer feature labels) to be appropriate for our scenario. Thus, we select MNB/Priors to power the DUALIST interface.

3.2 Tuning the Parameter α

A second question is how sensitive the accuracy of MNB/Priors+EM₁ is to the parameter α . To study this, we ran experiments varying α from 1 to 2^{12} , using different combinations of labeled instances and/or features (again using the simulated oracle and 10-fold cross-validation).

²Paired 2-tailed t -test, $p < 0.05$, correcting for multiple tests using the Bonferroni method.

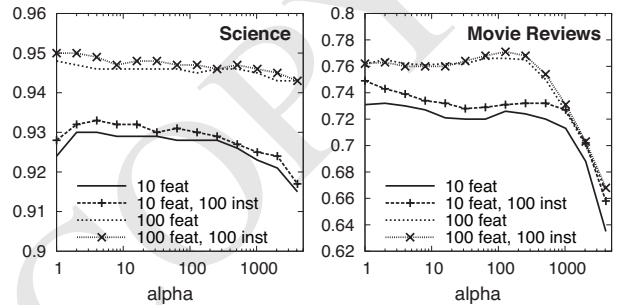


Figure 2: The effects of varying α on accuracy for two corpora, using differing amounts of training data (labeled instances and/or features). For clarity, vertical axes are scaled differently for each data set, and horizontal axes are plotted on a logarithmic scale. Classifier performance remains generally stable across data sets for $\alpha < 100$.

Figure 2 plots these results for two corpora (results qualitatively similar for the others). The first thing to note is that in all cases, accuracy is relatively stable for $\alpha < 100$, so tuning this value seems not to be a significant concern; we chose 50 for all other experiments in this paper. A second observation is that labeling 90 additional features improves accuracy much more than adding 100 labeled documents. This is encouraging, since labeling features (e.g., words) is known to be generally faster and easier for humans than entire labeling documents.

3.3 User Experiments

To evaluate our system in practice, we conducted a series of user experiments. This is in contrast to most previous work, which simulates active learning by using known document labels and feature labels from a simulated oracle (which do not necessarily behave like human oracles). We argue that this is a

necessary contribution, as it gives us a better sense of how well the approach actually works in practice, and also allows us to analyze behavioral results, which in turn may help inform future protocols for human interaction in active learning.

DUALIST is implemented as a Java web application and was deployed online. We used three different configurations: *active dual* (as in Figure 1, implementing everything from Section 2), *active instance* (instance queries only, no features), and a *passive instance* baseline (instances only, but selected at random). We also began by randomly selecting instances in the active configurations, until every class has at least one labeled instance or one labeled feature. $D = 2$ documents and $V = 100$ features were selected for each round of active learning.

We recruited five members of our research group to label three data sets using each configuration, in an order of their choosing. Users were allowed to spend a minute or two familiarizing themselves with the interface first, but received no training regarding it or the data sets. All experiments used a fixed 90% train, 10% test split for each corpus across all users, and annotators were not allowed to see the accuracy of the classifier they were training at any time. Each annotation action was timestamped and logged for analysis, and each experiment automatically terminated after six minutes.

Figure 3 shows learning curves, in terms of accuracy vs. annotation time, for each trial in the user study. The first thing to note is that the active dual configuration yields consistently better learning curves than either active or passive learning with instances alone, often getting within 90% of fully-supervised accuracy. The only two exceptions make interesting (and different) case studies. User 4 only provided four labeled features in the Movie Review corpus, which partially explains the similarity in performance to the instance-only cases. Moreover, these were manually-added features, i.e., he never answered any of the classifier’s feature *queries*, thus depriving the learner of the information it requested. User 5, on the other hand, never manually added features and only answered queries. With WebKB, however, he apparently found feature queries for the *course* label to be easier than the other classes, and 71% of all his feature labels came from that class (sometimes noisily, e.g., “instructor,” which can also

indicate *faculty* pages). This imbalance ultimately biased the learner toward the *course* label, which led to classification errors. These pathological cases represent potential pitfalls that could be alleviated with additional user studies and training. However, we note that the active dual interface is not particularly worse in these cases, it is simply not significantly better, as in the other 13 trials.

We found feature queries to be less costly than instances, which is consistent with findings in previous work (Raghavan et al., 2006; Druck et al., 2009). The least expensive actions in these experiments were labeling (mean 3.2 seconds) and unlabeled (1.8s) features, while manually adding new features took only slightly longer (5.9s). The most expensive actions were labeling (10.8s) and ignoring³ (9.9s) instance queries. Interestingly, we observed that the human annotators spend most of their time during the first three minutes performing feature-labeling actions (■■■■), then switching to more instance-labeling activity for the final three minutes of the experiment (■■■■). It may be that the active learner exhausts the most salient terms for the task after three minutes, so the user begins to focus on more interpretable instances. However, more study (and longer annotation periods) are warranted to better understand this phenomenon, which might suggest further user interface improvements.

We also saw surprising trends in annotation quality. In active settings, users made an average of one instance-labeling error per trial (relative to the gold-standard labels), but in the passive case this rose to 1.6, suggesting they are more accurate on the active queries. However, they also explicitly ignored more instances in the active dual condition (7.7) than either active instance (5.9) or passive (2.5), indicating that they find these queries more ambiguous. This seems reasonable, since these are the instances the classifier is least certain about. But if we look at the *time* users spent on these actions, they are much faster to label/ignore (9.7s/7.5s) in the active dual scenario than in the active instance (10.0s/10.7s) or passive (12.3s/15.4s) cases, which means they are being more efficient. The differences in time between dual and passive are statistically significant⁴.

³Selecting the “X” button for ambiguous documents.

⁴Kolmogorov-Smirnov test, $p < 0.01$.

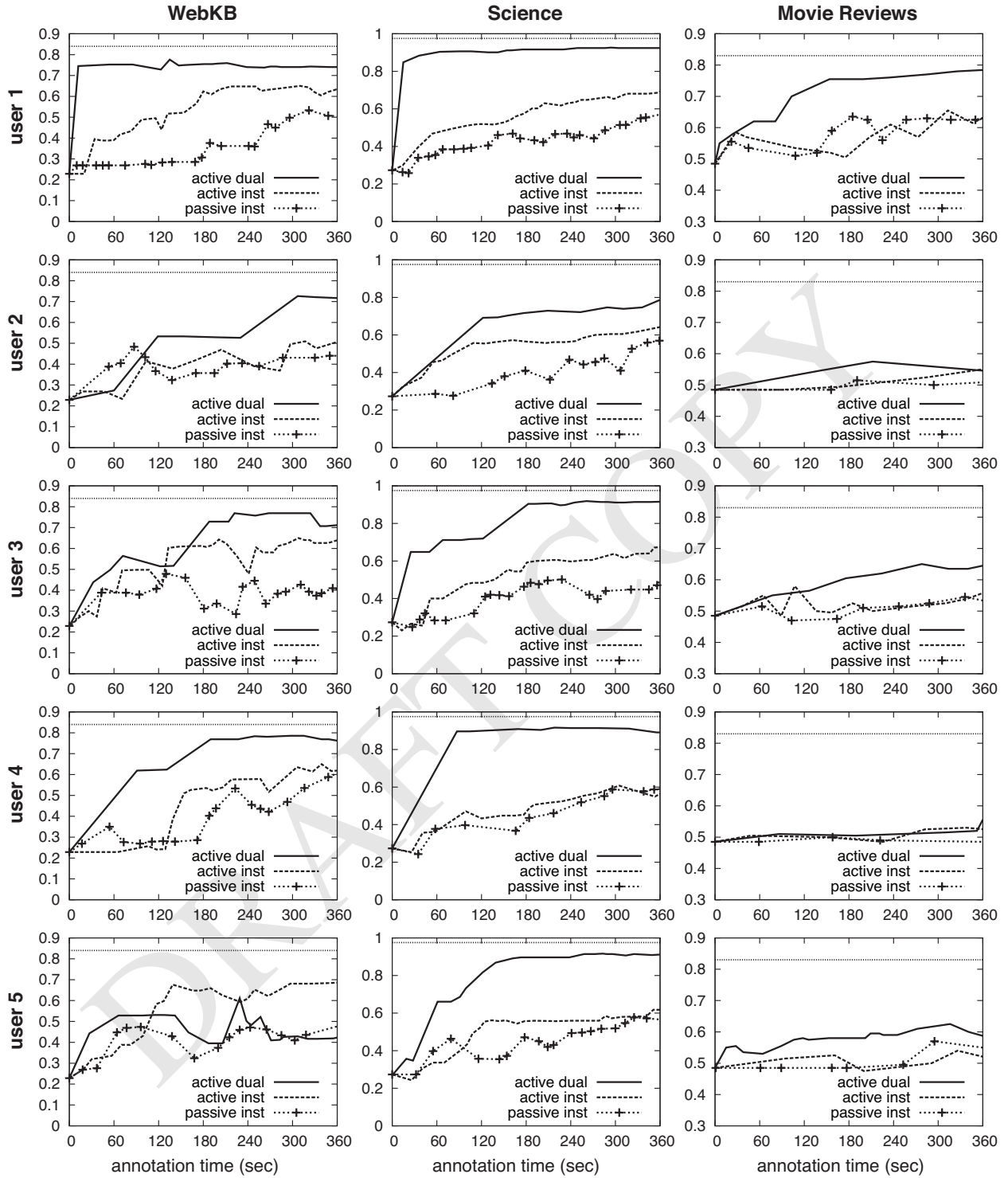


Figure 3: User experiments involving human annotators for text classification. Each row plots accuracy vs. time learning curves for a particular user (under all three experimental conditions) for each of the three corpora (one column per data set). For clarity, vertical axes are scaled differently for each corpus, but held constant across all users. The thin dashed lines at the top of each plot represents the idealized fully-supervised accuracy. Horizontal axes show labeling cost in terms of actual elapsed annotation time (in seconds).

3.4 Additional Use Cases

Here we discuss the application of DUALIST to a few other natural language processing tasks. This section is not meant to show its superiority for these tasks, but rather to demonstrate the flexibility and potential of our approach in a variety of problems in human language technology.

3.4.1 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the problem of determining which meaning of a word is being used in a particular context (e.g., “hard” in the sense of a difficult job vs. a marble floor). We asked one user to employ DUALIST for 10 minutes for each of three benchmark WSD corpora (Mohammad and Pedersen, 2004): Hard (3 senses), Line (6 senses), and Serve (4 senses). Each instance represents a sentence using the ambiguous word, and features are lowercased unigram and bigram terms from the sentence. The learned models’ prediction accuracies (on the sentences *not* labeled by the user) were: 83.0%, 78.4%, and 78.7% for Hard, Line, and Serve (respectively), which appears to be comparable to recent supervised learning results in the WSD literature on these data sets.

3.4.2 Information Extraction

DUALIST is also well-suited to a kind of large-scale information extraction known as semantic class learning: given a set of semantic categories and a very large unlabeled text corpus, learn to populate a knowledge base with words or phrases that belong to each class (Riloff and Jones, 1999; Carlson et al., 2010). For this task, we first processed 500 million English Web pages from the ClueWeb09 corpus⁵ by using a shallow parser. Then we represented noun phrases (e.g., “Al Gore,” “World Trade Organization,” “upholstery”) as instances, and used a vector of their co-occurrences with heuristic contextual patterns (e.g., “visit to *X*” or “*X*’s mission”) as well as a few orthographic patterns (e.g., capitalization, head nouns, affixes) as features. We filtered out instances or contexts that occurred fewer than 200 times in the corpus, resulting in 49,923 noun phrases and 87,760 features.

We then had a user annotate phrases and patterns into five semantic classes using DUALIST:

Class	Prec.	# Ext.	# Feat.	# Inst.
<i>person</i>	74.7	6,478	37	6
<i>location</i>	76.3	5,307	47	5
<i>organization</i>	59.7	4,613	51	7
<i>date/time</i>	85.7	494	51	12
<i>other</i>	–	32,882	13	115

Table 2: Summary of results using DUALIST for web-scale information extraction.

person, *location*, *organization*, *date/time*, and *other* (the background or null class). The user began by inserting simple hyponym patterns (Hearst, 1992) for their corresponding classes (e.g., “people such as *X*” for *person*, or “organizations like *X*” for *organization*) and proceeded from there for 20 minutes. Since there was no gold-standard for evaluation on this corpus, we randomly sampled 300 predicted extractions for each of the four non-null classes, and hired human evaluators using the Amazon Mechanical Turk service⁶ to estimate precision. Each instance was assigned to three evaluators, using majority vote to score for correctness.

Table 2 shows the estimated precision, total extracted instances, and the number of user-labeled features and instances for each class. While there is room for improvement (published results for this kind of task are often above 80% precision), it is worth noting that in this experiment the user did not provide any initial “seed examples” for each class, which is fairly common in semantic class learning. In practice, such additional seeding may help, as the active learner acquired 115 labeled instances for the null class, but fewer than a dozen for each non-null class (in the first 20 minutes).

3.4.3 Twitter Filtering and Sentiment Analysis

There is growing interest in language analysis for online social media services such as Twitter (Eisenstein et al., 2010; Petrović et al., 2010; Ritter et al., 2010), which allows users to broadcast short messages limited to 140 characters⁷. Two basic but interesting tasks in this domain are (1) language filtering and (2) sentiment classification, both of which are difficult because of the extreme brevity and informal use of language in the messages.

⁵<http://www.mturk.com>

⁶<http://boston.lti.cs.cmu.edu/Data/clueweb09/>

⁷<http://twitter.com>

Even though Twitter attempts to provide language metadata for its “tweets,” English is the default setting for most users, so 1/3 of English-tagged tweets are actually in a different language. Furthermore, the length constraints encourage acronyms, emphatic misspellings, and orthographic shortcuts even among English-speaking users, so many tweets in English actually contain no proper English words (e.g., “OMG ur sooo gr8!! #luvya”). This may render pure lexicon-based filters—and possibly n -gram language filters—ineffective.

To quickly build an English-language Twitter filter, we sampled 150,000 tweets from the Twitter Streaming API⁸ and asked an annotator spend 10 minutes with DUALIST labeling English and non-English messages and features. Features were represented as unigrams and bigrams without any stop-word filtering, but including some Twitter-specific features such as emoticons (text-based representations of facial expressions such as :) or :(used to convey feeling or tone), the presence of anonymized usernames (preceded by ‘@’) or URL links, and hashtags (compound words preceded by ‘#’ and used to label messages, e.g., “#loveit”). Following the same methodology as Section 3.4.2, we evaluated 300 random predictions using the Mechanical Turk service. The estimated accuracy of the trained filter was 85.2% (inter-annotator agreement among the evaluators was 94.3%).

We then took the 97,813 tweets predicted to be in English and used them as the corpus for a sentiment classifier, which attempts to predict the mood conveyed by the author of a piece of text (Liu, 2010). Using the same feature representation as the language filter, the annotator spent 20 minutes with DUALIST, labeling tweets and features into three mood classes: *positive*, *negative*, and *neutral*. The annotator began by labeling “smiley” and “frowny” emoticons, by which the active learner was able to discover some interesting domain-specific salient features, such as “cant wait” and “#win” for *positive* tweets, or “#tiredofthat” for *negative* tweets. Using a 300-instance Mechanical Turk evaluation, the estimated accuracy of the resulting sentiment classifier was 65.9% (inter-annotator agreement among the evaluators was only 77.4%).

4 Discussion and Future Work

We have presented DUALIST, a new type of dual-strategy annotation interface for semi-supervised active learning. The tool is available as an open-source project for the natural language processing community at large⁹. To support this dual-query interface, we developed a novel, fast, and practical semi-supervised learning algorithm, and demonstrated how users can use it to rapidly develop useful natural language systems for a variety of tasks. For several of these applications, the interactively-trained systems are able to achieve 90% of state-of-the-art performance after only a few minutes of labeling effort on the part of a human annotator.

This represents one of the first studies of an active learning system designed to compliment the needs of both learner and annotator. Future directions along these lines include user studies of annotation behaviors, which in turn might recommend new types of queries or other improvements to the user interface design. From a machine learning perspective, there is an open empirical question of how useful the labels actively gathered with DUALIST’s internal model (MNB) will be in training machine learning systems with different inductive biases, since the data are not sampled IID. Studies so far on this topic have yielded mixed results (Lewis and Catlett, 1994; Baldridge and Osborne, 2004).

Practically speaking, DUALIST is implemented to run on a single machine, and supports a few hundred thousand instances and features at interactive speeds (using typical modern consumer hardware). Distributed data storage (Chang et al., 2008) and parallelized learning algorithms (Chu et al., 2007) may help scale this approach well into the millions. Finally, modifying MNB to better cope with violated independence assumptions may be necessary for interesting language applications beyond those presented here. TAN-Trees (Friedman et al., 1997), for example, might be able to accomplish this while retaining speed and interactivity.

Acknowledgments

Thanks to members of Carnegie Mellon’s “Read the Web” research project for helpful discussions and

⁸http://dev.twitter.com/pages/streaming_api

⁹<http://code.google.com/p/dualist/>

participation in the user studies. This work is supported in part by DARPA (under contracts FA8750-08-1-0009 and AF8750-09-C-0179), the National Science Foundation (IIS-0968487), and Google.

References

- J. Attenberg, P. Melville, and F. Provost. 2010. A unified approach to active dual supervision for labeling features and examples. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*. Springer.
- J. Baldridge and M. Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL Press.
- A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1306–1313. AAAI Press.
- F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R.E. Gruber. 2008. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems*, 26(2):1–26.
- C.T. Chu, S.K. Kim, Y.A. Lin, Y. Yu, G. Bradski, A.Y. Ng, and K. Olukotun. 2007. Map-reduce for machine learning on multicore. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19, pages 281–288. MIT Press.
- M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. 1998. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 509–516. AAAI Press.
- G. Druck, G. Mann, and A. McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602. ACM Press.
- G. Druck, B. Settles, and A. McCallum. 2009. Active learning by labeling features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 81–90. ACL Press.
- J. Eisenstein, B. O'Connor, N.A. Smith, and E.P. Xing. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1277–1287. ACL Press.
- N. Friedman, D. Geiger, and M. Goldszmidt. 1997. Bayesian network classifiers. *Machine learning*, 29(2):131–163.
- S. Godbole, A. Harpale, S. Sarawagi, and S. Chakrabarti. 2004. Document classification through interactive supervision of document and term labels. In *Proceedings of the Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 185–196. Springer.
- M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Conference on Computational Linguistics (COLING)*, pages 539–545. ACL.
- D. Heckerman. 1995. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.
- K. Lang. 1995. Newsweeder: Learning to filter net-news. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339. Morgan Kaufmann.
- D. Lewis and J. Catlett. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann.
- B. Liu. 2010. Sentiment analysis and subjectivity. In N. Indurkha and F.J. Damerau, editors, *Handbook of Natural Language Processing*. CRC Press.
- A. McCallum and K. Nigam. 1998. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367. Morgan Kaufmann.
- P. Melville, W. Gryc, and R.D. Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1275–1284. ACM Press.
- S. Mohammad and T. Pedersen. 2004. Combining lexical and syntactic features for supervised word sense disambiguation. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 25–32. ACL Press.
- A.Y. Ng and M. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, pages 841–848. MIT Press.
- B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up: Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86. ACL Press.

- S. Petrović, M. Osborne, and V. Lavrenko. 2010. Streaming first story detection with application to twitter. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 181–189. ACL Press.
- H. Raghavan, O. Madani, and R. Jones. 2006. Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686.
- J.D. Rennie, L. Shih, J. Teevan, and D. Karger. 2003. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 285–295. Morgan Kaufmann.
- E. Riloff and R. Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 474–479. AAAI Press.
- A. Ritter, C. Cherry, and B. Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 172–180. ACL Press.
- F. Sebastiani. 2002. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.
- B. Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.