Document classification through interactive supervision on both document and term labels

Shantanu Godbole, Abhay Harpale, Sunita Sarawagi, and Soumen Chakrabarti

IIT Bombay
Powai, Mumbai, 400076, India
Contact: shantanu@it.iitb.ac.in

Abstract. Effective incorporation of human expertise, while exerting a low cognitive load, is a critical aspect of real-life text classification applications that is not adequately addressed by batch-supervised high-accuracy learners. Standard text classifiers are supervised in only one way: assigning labels to feature vectors derived from whole documents. They are thus deprived of the enormous wisdom that humans carry about the significance of words and phrases in context. We present HIClass, an interactive and exploratory labeling package that actively collects user opinion on feature representations and choices, as well as whole-document labels, while minimizing redundancy in the input sought. Preliminary experience suggests that, starting with essentially an unlabeled corpus, very little cognitive labor suffices to set up a meaningful labeled collection on which standard classifiers perform well.

1 Introduction

Motivated by applications like spam filtering, e-mail routing, Web directory maintenance, and news filtering, text classification has been researched extensively in recent years [10, 7, 13] by the machine learning community. State-of-the-art classifiers like Support Vector Machines (SVMs) now achieve up to 90% accuracy on well-known benchmarks. Almost all machine learning research related to text classification assumes some fixed, simple class of feature representation (such as bag-of-words). They also assume at least a partially labeled corpus. Statistical learners also depend on the deployment scenario to be reasonably related to the training population.

Many of these assumptions do not hold in real-life applications. Discrimination between labels can be difficult unless features are engineered and selected with extensive human knowledge, not left to a standard bag-of-words representation. Often, there is no labeled collection to start with. In fact, even the label set may not be specified up front, and must evolve with the administrator or user's understanding of the application. Several projects reported at the annual Operational Text Classification workshops [1] describe applications spanning law, journalism, libraries and scholarly publications in which automated, batch-mode techniques were not satisfactory; substantial human involvement was required before a suitable feature set, label system, labeled corpus, rule base, and resulting system accuracy were attained. However, not all the techniques used in commercial systems are publicly known, and few general principles can be derived from these systems.

There is much scope for building machine learning tools which engage the user in an active dialog to acquire human knowledge about features and labels. In the special case where supervision is available only as label assignments, a large body of research on *active learning* has provided some clear principles [4, 5] and strategies for maximum payoffs from the dialog. We wish to significantly extend the active learning paradigm to include both feature engineering and document labeling conversations, exploiting rapidly increasing computing power to give the user immediate feedback on her choices.

Our contributions: In this paper we present the design of a system HIClass (Hyper Interactive text Classification) for providing this tight interaction loop. We extend SVMs to naturally absorb human inputs in the form of feature engineering, term inclusion/exclusion and term and document labels. In the past, such actions were performed through ad hoc means and as a distinct processing step before classification construction. We make these more effective by (1) providing the user easy access to a rich variety of summaries about the learnt model, the input data and aggregate performance measures, (2) drawing the user's attention to terms, classes or documents in greatest need of inspection, and (3) helping the user assess the effect of every choice on the accuracy on test data.

Outline: We describe the HIClass workbench in Section 2 and review the design choices and various modes of user interaction. Section 3 describes our method of active learning on documents. Section 4 introduces the idea of active learning on terms. We report our experiences with the workbench and experimental results in Section 5. We review related work in Section 6 and conclude in Section 7.

2 The HIClass workbench for text classification

We present an overview of HIClass in Fig. 1. The lower layer shows the main data entities and main processing units in the system. There is a small pool of labeled documents (usually sampled into a training and test set) and a large unlabeled pool. A feature extractor turns the documents into feature vectors. Features are usually words, but the user can interactively refine features to be more complex; this is described next. The system can store and access by name multiple classifiers with their fitted parameters at any given time, assisting comparative analysis of parameters, performance on held-out data, and drill-down error diagnostics. The upper layer shows the prominent menus/modes in which a user can interact with the system. In the rest of this section, we will describe the important building blocks shown in Fig. 1.

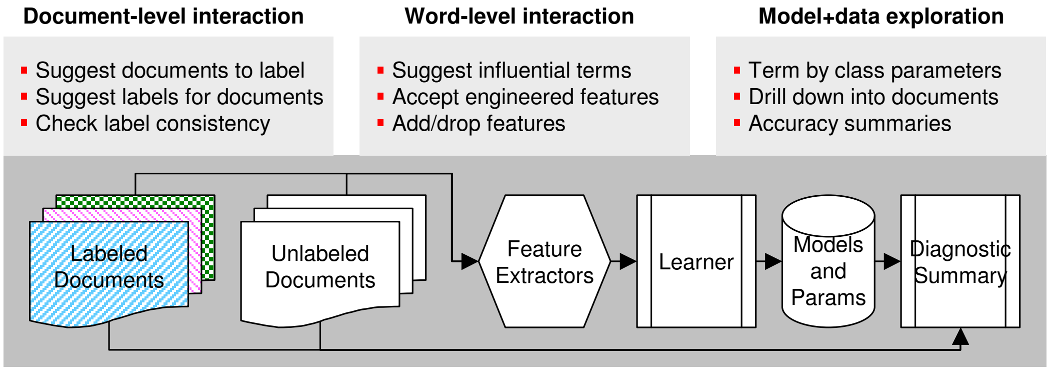


Fig. 1. The architecture of HIClass

2.1 Document and classification models

The first step of the design of HIClass is to choose a flexible classification model template that (1) suits state-of-the-art automated learners and (2) can be easily interpreted and tuned by the user.

A document is a bag of features. By default, and most often, features are words after minor processing like stemming and case-normalization. But the user can also (dynamically) define features to reflect domain knowledge. E.g., month names or currency names may be conflated into synthetic features. On the other hand, the user may notice harmful conflation between “blood bank” and “bank”, and define “blood bank” as a single compound feature. We will continue to use *term*, *word* and *feature* interchangeably where no confusion can result. With any given feature representation, a document is represented in the common vector space model, normalized to unit L_1 or L_2 norm.

Labeled documents can be associated with more than one class in general. HIClass supports **linear additive** classifier models, where each class c is associated with a set of weights $w_1^c, \ldots w_T^c$ corresponding to the T terms in a vocabulary. Each document is represented by a vector of non-negative weights $\boldsymbol{x} = (x_1, \ldots, x_T)$, each component corresponding to a feature. The classifier assigns a document all class labels c for which $w^c \cdot x + b_c \geq 0$ where b_c is a scalar per-class bias parameter. Since documents vectors have only non-negative components, both the magnitude and sign of components of w^c give natural interpretations of the salience of a term with respect to a class.

The linear additive model generalizes a number of widely-used classifiers, including naive Bayes (NB), maximum entropy, logistic regression, and support vector machines (SVMs). Here we focus on SVMs. Given documents d_i with labels $y_i \in \{-1, +1\}$, a two-class linear SVM finds a vector $\mathbf{w}$ and a scalar constant b, such that for all documents $y_i(\mathbf{w}_c \cdot d_i + b) \geq 1$, and $||\mathbf{w}_c||$ is minimized.

When the application demands more than two classes, one can (1) rewrite the above optimization slightly, with one $\mathbf{w}$ vector per class, so that the discriminant $\mathbf{w}_{c_j} \cdot d_i + b_j$ is largest for the correct class c_j; or (2) build an ensemble of SVMs, each playing off one class against another ("one-vs-one"), and assigning the document to the class that wins the largest number of matches; or (3) build an ensemble of SVMs, as many as there are classes, each predicting an yes/no label for its corresponding class ("one-vs-rest" or "one-vs-others"). In practice, all these approaches are comparable in accuracy [6]. We use one-vs-others as it is easily extended to make multi-labeled prediction and is efficient.

2.2 Exploration of data/models/performance summaries

HIClass facilitates the user to view the trained classifier scores, aggregate as well as drill-down statistics about terms, documents and classes, and accuracy measures on user-chosen corpus subsets.

After building an initial classifier using the starting labeled set L, the user can view the learnt model as a matrix of class-by-term scores. Simple inspection of term-class scores in this OLAP-like tool enables the expert to propose changes to per-class classification models like including, excluding and ignoring certain terms for certain classes. The user-interface is tightly coupled to allow easy movement from a term-centric to a class-centric analysis. In either of the term/class-centric interface, the user can see a projection of documents which contain particular terms contained in various classes, with proper term highlighting.

With every proposed change, the user can study the impact of the change by observing its performance on the test dataset. The user can inspect graphs for the accuracy of the whole system through iterations. Micro/macro-averaged precision and recall for the whole system as well as per-class precision and recall statistics are available. The user can identify classes which are hampering the overall performance of the system. The user can then concentrate on this class further or choose to add more labeled documents actively. A confusion matrix view could reveal two classes that confuse a lot with each other and the user can visualize/modify discriminating terms by exploring the results of a one-vs-one SVM on these two classes.

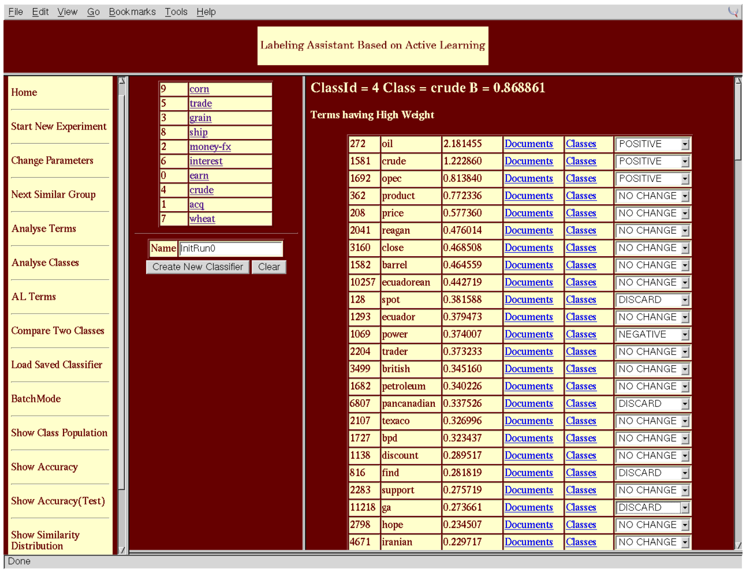
Various aggregate statistics of the data can also be seen at any active iteration. Per-class population, similarity distribution and uncertainty distribution help the user get an idea of the state of the system.

2.3 Feature Engineering

Fast evaluation over a variety of test data enables a user to easily identify limitations of a trained model and perhaps the associated feature set. Consider an example of the well-known Reuters-21578 dataset in Figure 2 which shows a few example rows and columns from the score matrix of a trained SVM model.

Most users, on inspection of the set of scores, will be able to propose a number of modifications to the classifiers. Some of these modifications may not impact performance on the available test set but it could be beneficial in improving the robustness and performance of the classifier on future unseen instances. Close inspection of some of the terms shown to have a high positive weight for the class *crude* in Figure 2 reveals that

– “Reagan” is found to be a positive indicator of the class *crude* though proper names should be identified. “Ecuador” and “Ecuadorean” show the insufficiency of the stemming algorithm used.
– Another frequent phenomenon is the inadequate combination of words. “World bank” and “Buenos Aires” should always occur together as a bi-gram; “Union”, a high weight term for *crude* should be associated with “Pacific Union” in *crude* and “Soviet Union” in other classes.
– Aggregation: Month names, currencies, date formats, proper nouns should be recognized and grouped into appropriate aggregate terms.



Labeling Assistant Based on Active Learning

9	corn
5	trade
3	grain
8	ship
2	money-fx
6	interest
0	earn
4	crude
1	acq
7	wheat

Name: InitRun0

ClassId = 4 Class = crude B = 0.868861

Terms having High Weight

272	oil	2.181455	Documents	Classes	POSITIVE
1581	crude	1.222860	Documents	Classes	POSITIVE
1692	opec	0.813840	Documents	Classes	POSITIVE
362	product	0.772336	Documents	Classes	NO CHANGE
208	price	0.577360	Documents	Classes	NO CHANGE
2041	reagan	0.476014	Documents	Classes	NO CHANGE
3160	close	0.468508	Documents	Classes	NO CHANGE
1582	barrel	0.464559	Documents	Classes	NO CHANGE
10257	ecuadorean	0.442719	Documents	Classes	NO CHANGE
128	spot	0.381588	Documents	Classes	DISCARD
1293	ecuador	0.379473	Documents	Classes	NO CHANGE
1069	power	0.374007	Documents	Classes	NEGATIVE
2204	trader	0.373233	Documents	Classes	NO CHANGE
3499	british	0.345160	Documents	Classes	NO CHANGE
1682	petroleum	0.340226	Documents	Classes	NO CHANGE
6807	pancanadian	0.337526	Documents	Classes	DISCARD
2107	texaco	0.326996	Documents	Classes	NO CHANGE
1727	bpd	0.323437	Documents	Classes	NO CHANGE
1138	discount	0.289517	Documents	Classes	NO CHANGE
816	find	0.281819	Documents	Classes	DISCARD
2283	support	0.275719	Documents	Classes	NO CHANGE
11218	ga	0.273661	Documents	Classes	DISCARD
2798	hope	0.234507	Documents	Classes	NO CHANGE
4671	iranian	0.229717	Documents	Classes	NO CHANGE

Fig. 2. Reuters started with 10 documents in each class. Term weights for the class 'crude' are shown in the OLAP-like tool to inspect documents, classes and terms.

2.4 Document labeling assistant

When unlabeled data is abundant and labeled data is limited, a user can choose to add labels to some of the unlabeled documents. Active learning has proven to be highly effective in interactively choosing documents for labeling so that the total number of documents to be labeled is minimized. However, this number may not be an accurate indicator of cognitive load on the user. In the end, we would like to minimize the "think time" of the user while maximizing learning rate. To that end, our system assists the user in the labeling task through a number of mechanisms:

- We group together related documents for labeling so that the user can **bulk-label** a number of documents in one shot.
- We provide a set of suggested class labels and for each suggested class the user can inspect the set of documents in it most similar to the document to be labeled.
- Document labeling is a complicated task and sometimes users make mistakes. We have a built-in **conflict checker** module that checks for gross violation in the consistency of the labeled set and shows them to the user.

Detailed mechanisms of performing the above tasks appear in Section 3.

2.5 Term-level active learning

The high accuracy of linear SVMs at text classification [7] suggests that the class membership decision depends on the combination of “soft” evidence from a class-conditional subset of the corpus vocabulary. E.g., high rate of occurrence of one or more of the words *wicket*, *run*, *stump*, and *ball* leads us to believe a document is about (the game) cricket.

Given enough training documents, good classifiers can learn the importance of these terms. But in the initial stages of bootstrapping a labeled corpus, it is far more natural for the user to directly specify these important features as being positively associated with the class “cricket”, rather than scan long documents for the trigger words to be able to assign yes/no labels to whole documents.

In HIClass we allow users to label terms with classes just like documents. We expect the cognitive load of labeling terms to be lower because the user does not have to waste time reading long documents where most of the words are either clearly associated with a class or are irrelevant. HIClass helps a user in spotting such terms by doing active learning on terms. We elaborate on this in Section 4.

3 Active learning on documents

The system starts with a small training pool of labeled documents L and a large pool of unlabeled documents U. Assume that the number of class labels is k and that documents can be assigned multiple labels. We train k one-vs-others SVMs on L.

Our goal during active learning is to pick some unlabeled documents about whose predictions the classifier is *most uncertain*. Various measures are used for calculating uncertainty in the literature for SVMs in text classification setting [12]. However, these assume binary, single-labeled documents. We extend these to the multi-labeled setting as described next.

3.1 Uncertainty

Each unlabeled document gets k discriminant values, one from each SVM in the one-vs-others ensemble. We arrange these values on the number line, and find the largest gap between adjacent values. A reasonable policy for multilabel classification using one-vs-others SVMs is that discriminant values to the right of the gap (larger values) correspond to SVMs that should be assigned a positive label to the document and the rest should be negative.

We need this policy because, in our experience with one-vs-others ensembles, as many as 30% of documents may be labeled negative by all members of the ensemble. For single label classification, it is common to pick the maximum discriminant even if it is negative. Our policy may be regarded as an extension of this heuristic to predict multiple labels.

With this policy, we declare that document to be most uncertain whose this largest gap is the smallest among all documents. When documents are restricted to have one label, this reduces to defining certainty (confidence) in terms of the gap between two highest scores.

3.2 Bulk-labeling

The user could label these uncertain documents one by one. But experience suggests that we can do better: often, many of these documents are quite similar, and if we could present tight clusters that the user can label all at once, we can reduce the cognitive load on the user and speed up the interaction.

We pick the u most uncertain documents and compute pairwise vector-space similarity between documents in the uncertain set, and prepare for the user a cluster/subset of fixed size (set by the parameter s) that has the largest sum of pairwise similarities.

When showing these uncertain clusters to the user, we also provide an ordered list of suggested labels. The ordering is created by taking the centroid of each uncertain cluster and finding its similarity to the k centroids of positive training data of the k classes.

(An alternative is to use the existing classifier itself to propose suggestions based on the confidence with which the documents in the uncertain cluster are classified into various classes. However, we feel keeping the same suggestion list for all documents in each uncertain cluster reduces the cognitive load on the user. Also, empirically we found in the initial stages this provides better suggestion than the SVMs.)

The user provides feedback to the system by labeling all documents in an uncertain cluster in one shot. The labeled documents are inspected by a conflict check module for consistency. We defer discussion of this topic due to lack of space. Once the user confirms the labels, the newly labeled documents are removed from U and added to L. The system then iterates back to re-training the SVM ensemble.

```
Start with a labeled pool $L$ and an unlabeled pool $U$.
while user wants to continue with active labeling do
  Train a $A$-vs-not$A$ SVM ensemble on $T$
  Calculate uncertainty on all documents in $U$:
  for all documents $d \in U$ do
    Get $k$ scores by applying the $k$ SVMs to $d$. Find the largest gap in score values.
  end for
  Sort the $|U|$ gaps in ascending order and add top $u$ to the uncertain set.
  Select the $s$ most similar documents from top $u$
  Suggesting ranked list of labels for the group $s$:
  for all $k$ classes do
    Find similarity between centroid of $s$ and centroid of positive training data of class $k$
  end for
  Sort these distances in a suggested list of classes
  Present $s$ and the ranked list of $k$ suggestions to the user for active labeling
  Accept multi-labeled suggestions for all documents in $s$. Check for conflicts
  Add these $s$ documents to $L$ with user provided labels and remove from $U$
end while
```

Fig. 3. The algorithm for active learning on documents

4 Active learning involving terms

As mentioned in Section 2.5, users generally find it easier to bootstrap the labeled set using trigger terms (that they already know) rather than start outright with a tedious scrutiny of lengthy documents for the known triggers. We first demonstrate this with a concrete example from the Reuters dataset.

Num labeled=1		Num labeled=50	
Term	w	Term	w
forecast	0.40	rate	2.08
bank	0.29	fe	1.97
noon	0.20	pct	1.65
account	0.20	market	1.26
oper	0.14	custom	1.01
market	0.14	interest	0.92
england	0.09	forecast	0.92
		stg	0.87
		bank	0.83

We trained two SVMs using the *interest* class in Reuters. One SVM was given one positive example labeled *interest* and one negative example from each of the other classes. The other SVM was given 50 documents with positive *interest* labels, and 50 negative examples from each of the other classes. Each SVM estimated a weight vector. For each SVM, we report some terms corresponding to the maximum weight components in the table.

The SVM that uses more data elicits terms like “rate” “fe” (foreign exchange), “pct” (percent), and “interest”: terms that a user can readily recognize as being positively associated with the label *interest*. But note that the SVM that uses very small amounts of training data cannot gather this evidence on its own, from document labels alone.

We argue that the user already knows many such trigger terms, and should not need to go through a document scrutiny and labeling process to make these important triggers be known to the system. Instead, we allow a direct process of proposing trigger terms within the additive linear framework. We believe such manual addition of terms will be most useful in the initial phases to bootstrap a starting classifier which is subsequently strengthened using document-level active learning.

We propose a mechanism analogous to active learning on documents to help a user spot such terms. The SVM treats labeled terms as mini-documents whose vector representation has a 1 at the term’s position and 0 everywhere else, thus having length 1 like regular documents

We develop a criterion for term active learning that is based on the theoretically optimum criterion of minimizing uncertainty on the unlabeled set but avoids the exhaustive approach required to implement it [4, 12, 5] by exploiting the special nature of single-term documents.

Consider adding a term t whose current weight is w_t in the trained SVM. For terms not in any of the labeled documents $w_t = 0$. Suppose we add t as a “mini-document” with the user-assigned label y_t. Let the new SVM weight vector be w'. Since the term t is a mini-document whose vector has $x_t = 1$ and $\forall t' \neq t, x_{t'} = 0$, we can assume that in the new w' only w_t is changed to a new w'_t and no other $w_{t'}$ is affected. This is particularly true for terms that do not already appear in the labeled set. Also, we know from the formulation of the SVM that for the term, $y_t(w'_t + b) \geq 1$. If the current w_t is such that $|w_t + b| > 1$ then adding t will probably not have any affect. So we consider only those ts where $|w_t + b| < 1$. Adding t with a label $+1$ will enforce $w'_t + b = 1$ i.e., $w'_t = 1 - b$ and with a label of -1 will make it $w'_t = -1 - b$. For each possible value of $y_t = c$, we get a new value of $w'_t(c)$. Thus we can directly compute the new uncertainty of each unlabeled document x by computing the *change* in the distance from separator value as $(w'_t(c) - w_t)x_t$. This gives us a way to compute the total uncertainty over the unlabeled set without retraining a SVM for each candidate term. The final algorithm is given in Fig. 4.

```
for all unlabeled documents $x^i \in U$ do
  Compute current distance from separator $d_i$
end for
for all term $t$ with $|w_t + b| < 1$ do
  for all class labels $c$ do
    Calculate $Pr(c, t)$: the probability that $t$ will be labeled with $c$
    $\mathrm{Pr}(c, t)$ = fraction of all docs that contain $t$ and are predicted $c$
    Estimate the new $w'_t$ as $(1 - b)$ when $c = 1$ and as $(-1 - b)$ when $c = -1$
    Compute the new distance for each unlabeled doc $x$ as $(w'_t - w_t)x_t + d_i$
    Compute uncertainty $U(c, t)$ with respect to the new distances
  end for
  $\mathrm{score}(t) = \sum_c U(c, t) \,\mathrm{Pr}(c, t)$
end for
Choose some terms with low value of scores
```

Fig. 4. The algorithm for term-based active learning

5 Experimental study

We have experimented with several text classifications tasks ranging from well-established benchmarks like Reuters-21578 and 20-newsgroups to more noisy classification tasks chosen from Web directories [11]. It is difficult to quantify the many ways in which HIClass is useful. Therefore we pick a few measures like the benefits of active learning with terms and document to report as performance numbers.

HIClass consists of roughly 5000 lines of C++ code for the backend and 1000 lines of PHP scripts to manage frontend user interactions. The frontend is a web browser, readily available on any user's desktop. XML is used to pass messages between the frontend and the server backend. LibSVM [3] is used as the underlying SVM classifier. Porter stemming and the SMART stoplist is used upon user preference.

All our development and experiments were done on a dual-processor P3 server running Debian Linux and with 2GB RAM. Due to space limitations we report numbers for fixed settings of some of our system parameters. Unless otherwise stated, the number of initial documents per class is set to 1, the number of documents selected for bulk labeling is 5 and the number of uncertain documents over which we pick similar clusters in the algorithm of Section 3 is set to 75.

5.1 Document-level active learning

We now show how active learning on documents can reduce the number of documents for which the user needs to provide labels in a multiclass,

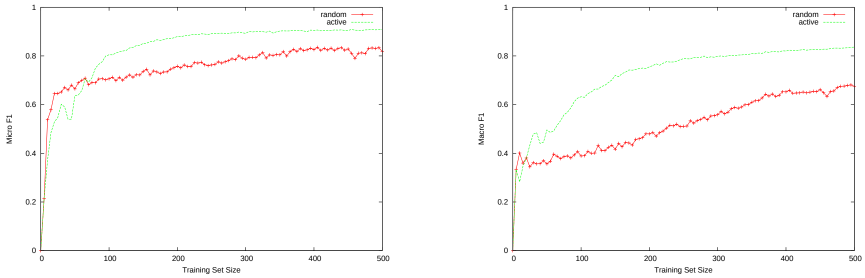


Fig. 5. Reuters-21578 - Micro and Macro-averaged F1 on held-out test data while increasing training set size, randomly versus using document level active learning.

multi-labeled settting. We started with one document in each class and added 5 documents in each round. All graphs are averaged over 30 random runs. Fig. 5 compares the micro and macro averaged $F1$, of selecting 5 documents per round using active learning and using random selection for Reuters (other datasets omitted due to lack of space). We see that active learning outperforms randomly adding documents to L and reaches its peak accuracy faster.

5.2 Reducing labeling effort

We next show the effectiveness of the two techniques that we proposed in Section 3 for reducing the effort spent for labeling a document. For lack of space we only show results with Reuters in this sub-section; results with other datasets were similar.

Quality of Suggestions We quantify the quality of suggestions provided to the user by the average rank of the true labels in the suggested list. We see in Figure 6 that even in the initial stages of active learning the true classes on an average are within rank 4 whereas the total number of possible classes is 20 for this dataset. We also see that the suggestions with u (the uncertain set size) fixed at 75 are better than at 10 as expected.

Bulk-labeling We quantify the benefit of bulk-labeling by measuring **inverse similarity**, defined as the number of true distinct labels in a batch of s documents as a fraction of the total number of label assignments in this batch. For example, if $s = 5$ and each document in a batch has one label and all of them are the same, then the inverse similarity is $\frac{1}{5}$.

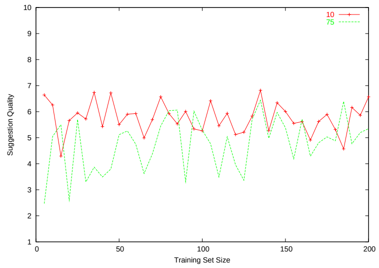


Fig. 6. Reuters - Quality of suggestion measured as the rank at which correct labels are found in the suggested labels

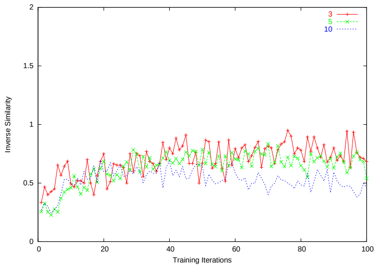


Fig. 7. Reuters - Benefits of bulk-labeling measured as inverse similarity defined in section 5.2

It is reasonable to assume that the cognitive load of labeling is proportional to the number of distinct labels that the user has to assign. Subject to this assumption, Fig. 7 establishes that our chosen set of similar documents reduce cognitive load by a factor of 2. The benefits are higher in the initial stages of active learning because then there are several documents with high uncertainty to choose from. Also, as we increase the number of documents per batch, the benefits get larger.

Of course, we cannot set s to be very high because there is a tradeoff between *reducing effort per label* by bulk labeling similar documents and *increasing number of labels* by possibly including redundant documents per batch. If we calculated labeling cost in terms of *number of documents* to be labeled, the optimum strategy is to label the most uncertain single document per batch. But the effort the user has to spend in deciding on the right label for rapidly changing document contexts will be high. The right tradeoff can only be obtained through experience and will vary with datasets and user's familarity with the data.

5.3 Term-level active learning

In these experiments our goal is to evaluate the efficacy of training with labeled terms. Therefore we conduct the following idealized experiment.

We take all available labeled documents for a class and train a one-vs-rest SVM for that class. All single-term documents that are predicted as positive or negative with very large margins (above $b/3$ in this experiment) are labeled with the predicted class and the rest are not labeled. We then start with a SVM trained initially with a single labeled document on each side and keep adding these collected labeled terms in order of their magnitude.

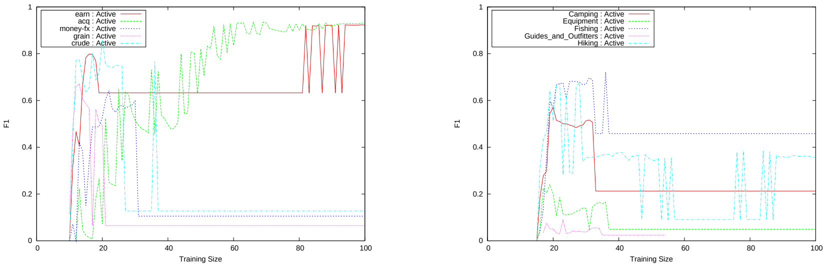


Fig. 8. Adding labeled terms in score order Reuters (left) and Outdoors (right)

In Fig. 8 we show the resulting accuracy on various classes of Reuters and the Outdoors dataset [11]. These graphs show some very interesting patterns. First, in almost all classes the addition of the first 20 terms achieves big jumps in accuracy for most classes and the accuracy goes from 5% with just one labeled document to 50-70% for Reuters. A similar jump is observed on the Outdoors dataset. However, unlike documents we cannot continue finding relevant terms and eventually unrelated terms can hurt accuracy. This confirmed our intuition that term-level active learning is best viewed as a bootstrapping technique which is later followed by document-level active learning. More experiments were terms to be added are selected via active learning are part of ongoing work.

6 Related Work

Most earlier work on applying active learning to text categorization [12, 9] assume a single binary SVM whereas our proposed scheme is for multiple one-vs-othersSVMs and for multi-labeled classification. Active learning has also recently been applied to the problem of selecting missing attributes of labeled instances whose values should be filled in by the user [8]. This differs from our setting of term active learning because our goal is to add terms as additional labeled instances. [2] uses term labeling for building lexicons of terms related to a concept. So the goal there is not to assign documents to categories but to exploit the co-occurrence patterns of terms in documents to categorize terms.

7 Conclusion

We have described HIClass, an interactive workbench for text classification which combines the cognitive power of humans with the power of automated learners to make statistically sound decisions. The system is based on active learning, starting with a small pool of labeled documents and a large pool of unlabeled documents. We introduce the novel concept of active learning on terms for text classification. We describe our OLAP-like interface for browsing the term-class matrix of the classifier cast as a linear additive model. The user can tune weights of terms in classes leading to better, more understandable classifiers. HIClass provides user continuous feedback on the state of the system, drawing her attention to classes, documents, and terms which would benefit by manual tuning.

In future work, we would like to engineer our system and make it more scalable to handle very large datasets. Another interesting avenue for further work is adding capabilities for adaptively evolving the label set as the classification process proceeds.

References

1. 3rd workshop on operational text classification. OTC 2003.
2. H. Avancini, A. Lavelli, B. Magnini, F. Sebastiani, and R. Zanoli. Expanding domain-specific lexicons by term categorization. In *SAC*, 2003.
3. C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines (version 2.31).
4. D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.
5. Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.
6. C. Hsu and C. Lin. A comparison of methods for multi-class support vector machines, 2001.
7. T. Joachims. Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98*
8. D. Lizotte, O. Madani, and R. Greiner. Budgeted learning of naive-bayes classifiers. In *UAI*, 2003.
9. A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. *Proceedings of ICML-98*
10. K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification, 1999.
11. S. Sarawagi, S. Chakrabarti, and S. Godbole. Cross-training: Learning probabilistic mappings between topics. In *SIGKDD*, 2003.
12. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, Nov. 2001.
13. J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. In *SIGIR*, 2003.