

When will Feature Feedback help? Quantifying the Complexity of Classification Problems

Hema Raghavan¹ and Omid Madani² and Rosie Jones²

¹ 140 Governor's Drive
University of Massachusetts
Amherst, MA 01002, USA
hema@cs.umass.edu

² Yahoo! Research 3333 Empire Avenue
Burbank CA 91504, USA,
{madani, jonesr}@yahoo-inc.com

Abstract. Supervised learning typically requires human effort to label a large number of training instances. Active learning strives to decrease the number of labeled training examples needed by actively engaging the learner and the human in an interactive process. Active learning has proven to be effective in many domains. With few training examples, past work has found that user prior knowledge on the importance of features, or interactive feature feedback, can guide the learner to converge faster, that is, with lower labeling costs. In this paper we aim to understand the kinds of problems for which such extra feedback are significantly beneficial. In other words, we ask what kind of problems can significantly benefit from interactive learning and whether for some problems the user has no choice but to engage in the tedious process of labeling many examples. Towards this goal, we define a set of four difficulty measures, 2 each of instance and feature complexity, for linear classification problems. These measures can efficiently be computed for real world problems for which linear classifiers are effective, such as text classification.

We quantify the difficulty of 358 text classification problems and 9 corpora using our measures, illustrating the spectrum of problems that exist in text classification in addition to quantifying results that have only been qualitatively discussed in the text classification literature. We verify the intimate relationship (a high positive correlation) between feature complexity and instance complexity using our measures. We then use these measures to understand when feature feedback is likely to be very useful. We observe that many problems in the commonly used data sets are of low to medium complexity, that is, only roughly 10s of well selected features are required to gain most of the maximum attained performance on such concepts. We find that learning these kinds of problems especially stands to benefit from feature feedback.

We note that our empirical difficulty measures and the rankings of problems and domains are of independent interest, beyond the active learning setting.

1 Introduction

Some concepts are easier to grasp by humans than others. For example, a human can probably grasp the concept of a “bird” with a few illustrative examples, or by a definition that lists the key properties of a bird (that it has feathers, a beak and so on). By

contrast, to learn the concept “art” a human probably needs to see many more examples and the key features can probably not be listed as easily as they could for “birds”. Much work has been done in the field of cognitive science in trying to understand why some concepts can be learned faster than others and we refer the interested reader to the works of Feldman [9], Chater [19] and others [28]. Feldman tried to characterize human error on concepts as a function of the Boolean complexity (the length of the shortest logically equivalent propositional formula) of a concept³. He found a ‘surprising’ (in his words) empirical ‘law’: the subjective difficulty of a concept in human learning is directly proportional to the Boolean complexity of the concept. In this work we wonder about the variability in concept difficulty in standard text classification tasks. We show that for a given learner and a set of concepts (categories in text) that can be learned by this learner, there exists a significant diversity in the difficulty of concepts in text. We ask whether some concepts are easier to learn than others. More specifically, we wonder whether some categories can be learned using a few training examples or features, while others may require many more examples and features before the concept is learned to the best of the learner’s ability. We define a set of measures that quantify the difficulty of concepts, illustrating the spectrum of problems that exist in text classification. In fact we too draw a conclusion similar to Feldman: concepts that can be learned using fewer examples can be described by a few well chosen features.

<i>wheat & farm</i>	→	WHEAT
<i>wheat & commodity</i>	→	WHEAT
<i>bushels & export</i>	→	WHEAT
<i>wheat & agriculture</i>	→	WHEAT
<i>wheat & tonnes</i>	→	WHEAT
<i>wheat & farm&¬soft</i>	→	WHEAT

Table 1. Induced rule set using the CONSTRUE system [1] for categorizing *wheat* documents in the Reuters data set. The induced rules result in 99% accuracy.

Given a learning algorithm, a set of features and a concept, there is some maximum achievable measure of categorization accuracy ($\leq 100\%$) that the learner can achieve in the limit. For example, even if the data is not exactly linearly separable, a linear SVM may be able to achieve some fairly reasonable and acceptable accuracy (often in the order of 90% for many text categorization problems) with adequate training data. Given such a set of concepts that are “almost linearly separable”, that is “learnable” by an advanced method such as a linear SVM, we ask how much training is adequate to attain nearly maximum achievable accuracy.

One view of concept complexity or difficulty may be one associated with the *value* of the maximum achievable accuracy, that is, a concept that cannot be learned to a desired degree of accuracy may be considered to be a difficult one. Studying difficulty from that perspective is important in itself, but is not the goal of this work. In this work

³ Example boolean expressions for a text classification task are shown in Table 1

we restrict ourselves to concepts that we know are ultimately sufficiently learnable by the chosen algorithm (SVMs) and ask how quickly they can be learned. The analogy in human learning would be with concepts taught at an elementary school level: they can all be learned if enough effort is put in by a student, yet some are easier (quicker to grasp) than others. On the other hand, some problems encountered at the graduate school level (classifying problems into P and NP complete categories for example) may be difficult in that they are not easy to solve and therefore less learnable.

We propose and explore a set of difficulty measures based on either the number of instances or the number of features needed to achieve approximately maximum accuracy achievable for the learner. Our instance complexity measures are designed to capture the number of training examples needed to attain nearly maximum achievable accuracy, when all the features are available. Thus, the training examples need not be selected at random; they can be intelligently picked. A problem for which training on a few well-picked instances is sufficient to arrive at the maximum achievable accuracy is a low instance complexity problem. Analogous to instance complexity, we define feature complexity with the goal of capturing the (approximately) minimum number of intelligently picked features needed to achieve nearly maximum possible accuracy, when ample training instances are available. If a concept can be described by a weighted combination of a few well selected features, it is considered to be of low feature complexity. We are therefore after capturing the inherent sample-size and feature-size complexity of a learning problem.

We study the relationship between the measures in text classification, a domain of high dimensionality with many relevant and irrelevant features. We find that instance complexity and feature complexity are highly positively correlated and yield similar rankings of problems and corpora (Section 5.1). This is consistent with the intuition that problems requiring large numbers of instances should require large numbers of features, and vice versa. This observation also provides evidence that our proposed measures indeed capture (approximately) the inherent feature and instance complexity of a problem. We benchmark 9 corpora and 358 text classification problems for their difficulty (Table 3), and analyze and discuss the reasons for the differences in complexity in several cases. Such knowledge can aid researchers and practitioners in assessing their own learning problems or in the selection of commonly used test data sets, and in anticipating learning performance. We find that many categorization problems are in the low to mid range complexity, that is, the number of intelligently picked features sufficient to obtain most of the accuracy, in terms of precision and recall, is in the 10s.

We began this work by asking why prior knowledge on features (e.g., [8, 22, 32, 26]) was more useful in accelerating learning for certain learning problems than others, and we considered whether there would be aspects of a problem that could explain or correlate well with the improvement from using feature knowledge. In Section 6 we describe how we use these measures to obtain insights into the kinds of text classification problems for which feature knowledge/feedback (in addition to document feedback) is especially useful. In particular, we find that problems with low to medium feature complexity stand to benefit most from feature feedback, and it is encouraging to see that many experimental problems fall in this range (Table 3 and Figure 8).

We begin by describing the data sets in Section 2. We describe our complexity measures in Section 3 and the methods that we use to instantiate our measures in Section 4. Sections 5 and 6 present our experiments and results. Section 7 presents a discussion of our methods and related work, and Section 8 concludes.

2 Data

We focus on text classification problems. We consider 9 corpora and 358 binary classification problems as shown in Table 2. In computing complexity for the Reuters-RCV1 corpus we only used the 23149 training documents for efficiency reasons [17].

Most corpora have topic-based category labels, except for three: (1) the Topic Detection and Tracking corpus that contains classes based on events. (2) the British National Corpus BNC corpus where the classes are based on genre. (3) The documents in the Enron corpus are emails categorized into folders by the recipient of the email.

For all data sets we used unigram features. For some of them we further added n -grams of features if these n -grams improved performance. All words in the text were stemmed and stop-words were removed using the rainbow toolkit [18]. Since we are only interested in measuring the difficulty of “learnable concepts”, we considered only those problems for which there was ample training data to achieve an acceptable level of performance (of above 75% Maximum F1) using a linear SVM. The last column in Table 2 lists the average maximum $F1$ obtained using a linear classifier and bag-of-word features trained on 90% of the data and tested on the remaining.

3 Measures of complexity

We now describe 4 measures of complexity – 2 each of instance and feature complexity. Given a “learnable concept” (or an “almost linearly separable concept”) with M labeled examples to estimate complexity from, each represented as an N dimensional vector, our complexity measures quantify the difficulty of learning by measuring how many of the M instances and N features are really required to learn a good classifier.

Consider a learning algorithm which is supplied with a set of training examples, ordered such that the most useful examples for learning are before the less useful ones. If only a few of these training instances are required for learning the task to high performance, we will say the task has low instance complexity. If a large number are required, we will say the task has high instance complexity.

Our instantiation of these instance complexity measures attempts to capture *roughly* how many of the best (most informative) instances for a given problem are needed in order to achieve performance close to that of a linear classifier that has access to all the features and ample training examples. In computing instance complexity we use active learning methods which give us an empirical upper bound on complexity. The tightness of the bound is dependent on the active learning method used. Similarly, our feature complexity measures quantify roughly how many of the most informative features are needed to achieve close to the best accuracy. Our feature complexity measures are also upper bounds on the true feature complexity, where the tightness of the bound is dependent on the feature selection method used. See Section 7 for further discussion.

Corpus	Domain	# instances	# features (N)	# topics	MaxF1
Reuters-21578	News-wire	9410	33378	10	0.874 (0.087)
Reuters-RCV1	News-wire	23149	47236	87	0.759(0.127)
Topic Detection Tracking(TDT)	News-wire and broadcast	67111	85436	10	0.918(0.001)
British National Corpus	News, journals etc.	2642	233288	15	0.774 (0.153)
Enron	E-mail folders	1971	711815	8	0.887(0.082)
20 Newsgroups	Newsgroup postings	19976	137728	20	0.851(0.007)
Industry Sector	Corporate web-pages	9565	69297	104	0.909(0.04)
TechTC-100	ODP hierarchy	149	18073	100	0.972(0.026)
WebKB	University websites	2101	28682	4	0.918(0.047)

Table 2. For all corpora except TechTC-100 there is a one one-versus-all binary classification problem. The TechTC-100 dataset consists of a 100 binary classification problems with about 149 documents in each and an average of 18073 features in each.

3.1 Instance Complexity Measures

Given a classification algorithm and a binary classification problem, there is some maximum achievable performance often under 100% in practice (Table 2). For accuracy performance we use the F1 score, which is the harmonic rank of precision and recall [27] (and see Section 4.4). We denote by $F1(p, q)$ the F1 score achieved by the learner when it has access to p instances and q features, $p \leq M$ and $q \leq N$. We assume the best performance is achieved when the learner has access to all the available instances and features, $F1(M, N)$, and not for example a subset of the features. This is a mild assumption for text classification problems, and in general linear classification problems, when one uses robust learners such as support vectors machines together with appropriate regularization.

In measuring the rate of learning we want to measure the minimum number of training examples (\hat{i}) needed to achieve the best performance for a given classifier. The brute-force way to find this minimum for a data set with M examples would require training the classifier for every possible subset of training examples, that is, 2^M times. The size of the minimum sized subset that gives performance close to the optimal performance is given as $\hat{i} = \min\{\operatorname{argmax}_i F1(i, N), i=1\dots 2^M\}$. This method, although most accurate, is time-consuming especially for large M . Instead we use active learning to give us an ordering on the instances and estimate an upper bound on \hat{i} using this ordering in the following way.

Active learning begins with 2 randomly selected instances, one in the positive and one in the negative class. The active learner learns a classifier based on this information and then intelligently chooses the next instance from a pool of unlabeled examples for the expert to label. The classifier is retrained and the process continues. We measure the performance, $F1(2^t, N)$ of the classifier after every 2^t iterations of active learning with t varying as $1, 2, \dots, \log_2 M$, where M is the total number of instances available for training. A performance curve for three problems in the 20 Newsgroups data set is shown in Figure 1. For the concepts – *graphics* and *ms-windows.misc*, the learner achieves the maximum attainable accuracy (0.70 F1) after seeing 2048 (2^{11}) examples. The value 2048 can be considered to be an upper bound on \hat{i} . For *sci.crypt*, the learner achieves its peak after seeing about 1024 examples, making it an easier concept (by our definition of complexity) than the other two. Each instance is chosen with the expectation that adding it to the training set will improve accuracy significantly. Since at each stage we are adding an example based on an estimate of its value to the training set, the bound is approximate. We can tighten the bound by providing the learning algorithm with as much information as possible: a large pool size for example. The advantage of using active learning is that the classifier needs to be trained only $O(M)$ times. How close this estimated complexity is to the true bound is dependent on the ability of the learner to leave out redundant instances in its training.

This simple measure of complexity is only an approximation to \hat{i} and a keen observer will note that the rate of convergence of the *ms-windows.misc* is initially higher than that of *comp.graphics*. It seems intuitive that *ms-windows.misc* should be considered to be less complex than *comp.graphics*. The approximate complexity value of 2048 estimated using active learning does not capture this learning rate. We factor in

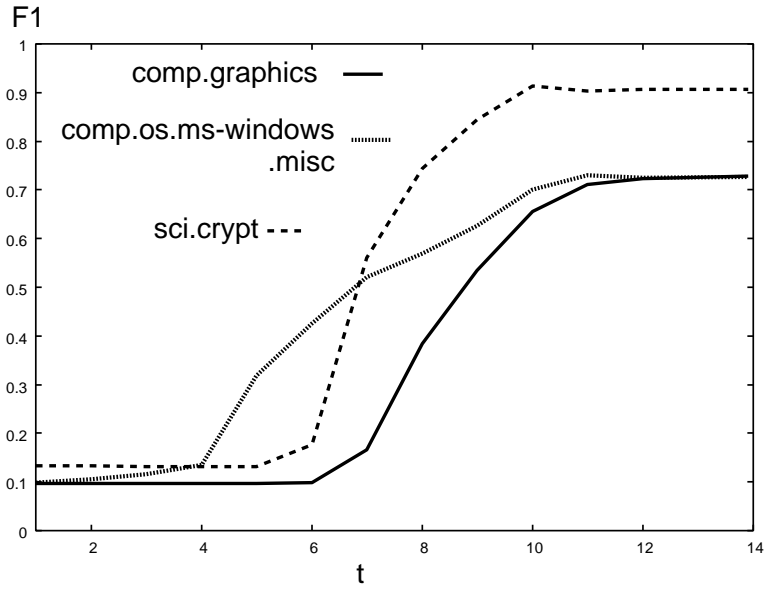


Fig. 1. Learning curves for a single classifier on 3 problems. The number of actively picked instances is 2^t , and F1 (Y-axis) is the harmonic mean of precision and recall.

the learning rate by considering the area under the learning curve computed as :

$$\text{AUC}_{\log} = \sum_{t=1}^{\log_2 M} F1(2^t, N)$$

This time we measure performance at exponentially increasing intervals, and compute the area under the learning curve, plotted with a logarithmic X-axis. AUC_{\log} implicitly gives a higher score to problems that converge more rapidly in the early stage of learning than later. To obtain a quantity that measures the rate of learning, we define the *active learning convergence profile* as follows:

$$p_{\text{al}} = \frac{\sum_{t=1}^{\log_2 M} F1(2^t, N)}{\log_2 M \times F1(M, N)} \quad (1)$$

p_{al} is the area under the normalized active learning curve (See Figure 2(a)), with a range between 0 and 1 and is independent of the number of instances needed for learning. Higher p_{al} implies faster convergence. The p_{al} values for the three problems in Figure 1 – *ms-windows.misc*, *comp.graphics* and *sci.crypt* are 0.61, 0.45 and 0.55 respectively. Note that even though the maximum accuracy achieved for *sci.crypt* is much higher (0.90 F1) than for the other two problems, the rate of active learning of *sci.crypt* is more similar to *comp.graphics*. The concept *ms-windows.misc* has the best rate of learning in the early stages. All these properties are captured by the p_{al} values.

We now describe the two instance complexity measures developed using the approximation to \hat{i} and p_{al} . For both measures, a higher value of complexity implies a more difficult problem.

1. Instance profile complexity, I_{pc} : This measure is simply the complement of the active learning convergence profile, and is given as $I_{pc} = 1 - p_{al}$. The active learning curve and hence the value of I_{pc} obtained is subject to the active learning algorithm and will be less than the ideal (theoretical best ordering of instances) case. Therefore, I_{pc} is an upper bound on the true complexity.

2. Instance complexity, C_i : I_{pc} only considers the rate of learning and does not contain any information about the number of instances needed to achieve the best performance. We therefore define $C_i = I_{pc} * n_i$ where n_i is the base 2 logarithm of the number of instances needed to achieve 95% of the best performance. We expect that n_i is an upper bound on $\log_2(\hat{i})$. We chose a threshold of 95%, rather than waiting for the curve to reach its peak, with the hope of capturing the point where most of the concept is learned. Usually, the rate of improvement at the final stages of learning, before the concept is fully learned, is very slow (see eg., [20]) with several thousands of instances contributing to a tiny improvement in performance, unnecessarily inflating the complexity score (See Figure 2(a)).

Using a log scale for n_i makes the scale like the Richter where an earthquake of magnitude 6 is significantly more intense than one of magnitude 5.

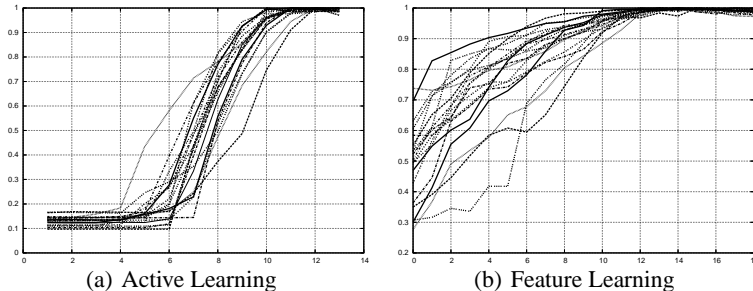


Fig. 2. Normalized learning curves (active learning and feature learning) for 20 Newsgroups.

3.2 Feature Complexity Measures

Our third and fourth measures attempt to capture the complexity of the problem in terms of the number of features needed to reach the best possible performance, when all the training instances are available. Again, instead of evaluating 2^N combinations of features, we estimate an approximation of the true feature complexity by using an oracle to learn a ranking of the features in the order of decreasing discriminative ability for a given classification problem. The oracle uses a large number of training documents and a feature selection criterion like information gain. We consider the performance

of the classifier constructed using k top ranking features. We plot a feature learning curve by plotting performance at exponentially increasing intervals of k . The normalized area under this feature learning curve, *the feature learning convergence profile*, p_{fl} is computed as follows:

$$p_{\text{fl}} = \frac{\sum_{k=1}^{\log_2 N} F1(M, 2^k)}{\log_2 N \times F1(M, N)} \quad (2)$$

Normalized feature learning curves for the 20 Newsgroups corpus are shown in Figure 2(b). The two feature complexity measures defined below are almost identical in intuition to the instance complexity measures.

1. Feature profile complexity, F_{pc} : *Feature profile complexity* (F_{pc}) is then defined as $F_{pc} = 1 - p_{\text{fl}}$. The computed value of F_{pc} is limited by the accuracy of the feature selection algorithm.

2. Feature complexity, C_f : Similar to C_i , we define $C_f = F_{pc} * n_f$, where n_f is the base 2 logarithm of the number of features in the feature learning curve needed to achieve 95% of the best performance. How good the estimate of the true feature complexity obtained this way is dependent on the feature selection algorithm used.

4 Methods

We first describe the two linear classifiers of choice: perceptrons and support vector machines (SVMs). We then describe our choice of active learning methods, followed by our choice of feature selection techniques.

4.1 Classifiers

A linear classifier is usually sufficient for text classification in part due to the very high dimensionality of text. The simplest algorithm for a linear classifier is the **Perceptron** algorithm [23]. The perceptron learns a linear function of the form $f(X_i) = w \cdot X_i + b$. It is a mistake-driven, incremental algorithm: when a new training example is added, the weight vector is adjusted only if it is misclassified. Therefore, the classifier needs to be trained less than (or equal to) $|\mathcal{T}|$ times, where \mathcal{T} is the training set. The correction to the weight vector is a simple adjustment of the form $w_i = w_i + \eta Y_i X_i$, for every instance X_i that is misclassified. Y_i is the true label of X_i ($Y_i \in \{+1, -1\}$). The parameter η called the learning rate.

Support vector machines are learning techniques that have gained much popularity in the recent past [31] and particularly so for text classification [14]. For many linearly separable problems there can be more than one hyperplane that separates the data (see Figure 3(a)). A support vector machine (SVM) on the other hand, is a maximum margin classifier that tries to find the hyperplane that results in a maximal separation of the two classes (See Figure 3(b)). The margin is the distance between the positive example closest to the hyperplane and the negative example closest to the hyperplane, with the distance being measured along a line perpendicular to the hyperplane. Although the motive for a margin that is maximal seems intuitive, it is also well motivated by the Vapnik-Chervonenkis theory that states that such a hyperplane minimizes expected test error [31]. Support vector machines have been proven to be effective in many domains, and especially so for text classification and filtering [14, 5].

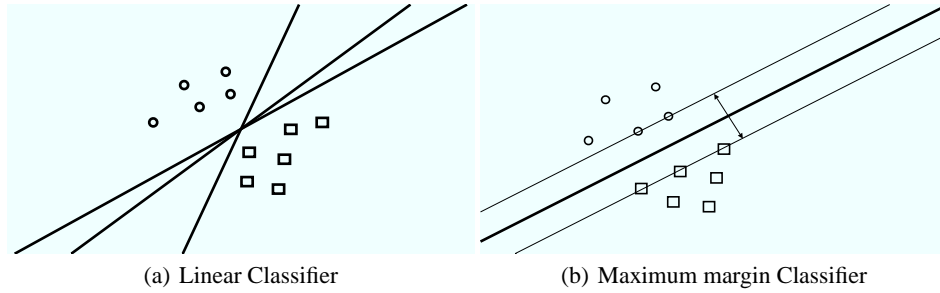


Fig. 3. Linear classifiers.

4.2 Active Learning

One method for instance selection is **uncertainty sampling** [16]. Uncertainty sampling is a simple, efficient, and commonly used type of active learning in which the example that the user (teacher) is queried on is the unlabeled instance that the classifier is least confident about. When the classifier is an SVM, unlabeled instances closest to the margin are chosen as queries [30]. If an uncertain instance lies exactly on the hyperplane it results in a reduction of the version space in exactly half [30]. If we can keep querying the user on examples that lie on the hyperplane we can decrease the number of training examples exponentially (by reducing the version space by half with each query) when compared to the case when the training data is obtained through random sampling. In reality, there may not be an example exactly on the hyperplane at each round of active learning, and hence we do not see the theoretical exponential decrease, but nevertheless for many text classification problems, uncertainty sampling is significantly better than random sampling [16]. While many other selective sampling methods have been proposed, uncertainty sampling carries the advantages of simplicity and efficiency, and we expect that for our rough measures of complexity in the text domain, uncertainty sampling is adequate. We compare against random sampling in our experiments.

Using uncertainty sampling with SVMs would involve retraining the SVM $O(M)$ times, which can be very time consuming as SVM training can involve quadratic optimization. Therefore, when we use SVM uncertainty sampling to compute p_{al} , we plot the learning curve only up to 1024 instances. To plot the complete active learning curve we use another learning method – **a committee of perceptrons** [6]. The perceptron algorithm being mistake-driven and online, takes less than time in each retraining than the SVM. Of course, active learning using perceptrons may not be as effective as SVM uncertainty sampling, and in particular, the numbers that we obtain using different methods can be somewhat different. However, we find that the ranking of the problems by their complexity computed using the perceptron committee is almost identical to the ranking obtained using SVM uncertainty sampling (refer Figure 7).

4.3 Feature Selection

Information gain is a simple, efficient and commonly used measure for ranking features that has been found to be quite effective [27, 4]. Information gain is given as:

$$IG = \sum_{c \in \{-1, +1\}} \sum_{\tau \in \{0, 1\}} P(c, \tau) \log \frac{P(c, \tau)}{P(c)P(\tau)}$$

where c denotes the class label (+1 or -1), and τ is 0 or 1 indicating the presence or absence of a feature respectively.

Information gain is our primary feature selection method. However, information gain does not ignore redundant features, and more generally it does not address feature dependencies (a feature’s quality is computed independent of others). This can inflate our feature complexity scores. So we also experimented with **SVM LARS** [15], a new and effective forward selection technique for feature selection. Given that it is a forward selection technique, LARS ignores highly correlated features in its feature selection, something information gain does not do. Therefore, we expect that LARS would capture the true feature complexity better by eliminating redundant features. However, SVM LARS has a relatively high running time and we use it only in a limited way by computing p_{fl} by plotting the feature learning curve only up to 1024 features. When we use information gain we are able to plot the entire learning curve.

4.4 Computing Performance

Each time we compute $F1(2^t, 2^k)$ in equations 1 and 2, our aim is to find the best possible performance with a classifier trained on 2^t examples and 2^k features. We hope that by using active learning with a large pool, and feature selection using a large training set, we obtain a fairly accurate estimate of this best classifier. The better the active learning and feature selection methods, the tighter the bound. Our experience with SVMs showed that with few training examples, much of the error is in a poor estimation of b . Hence, to obtain an even tighter bound, we sweep through all values of b and use that b for which the F_1 is maximum on the test set. We call this quantity **MaxF1**. In fact in Table 2, the last column lists the Max F1 values obtained with a 90-10 training-test split of the corpus.

5 Results

We made a number of simplifying assumptions in instantiating our difficulty techniques. Uncertainty sampling and feature ordering and selection using information gain are imperfect methods. Different learning algorithms can have close but still different accuracies. Finally, there are other potentially relevant factors to complexity that we have ignored. Potentially relevant aspects of a problem include the average length of documents, the size of the feature set (N), and the proportion of the positive documents in the corpus. In this section and the next, we explore the utility of our measures. We describe the results of using our complexity measures on the 358 problems described in Table 2.

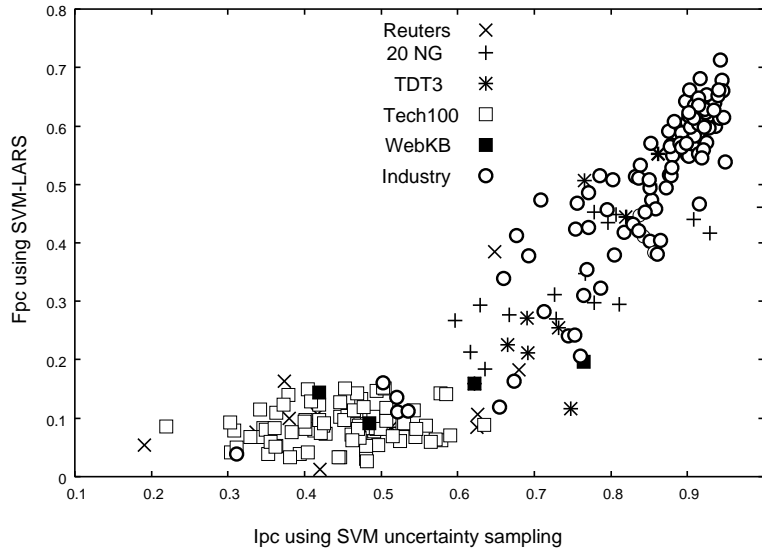


Fig. 4. Correlation between I_{pc} and F_{pc} using SVM and LARS. Correlation of instance complexity and feature complexity is independent of methods used to compute the two.

5.1 Correlation of Instance Complexity and Feature Complexity

Intuitively, good instance and feature complexity measures should correlate well: a problem that requires a large number of instances, must require finding the right mix of weights for a relatively large number of features, and vice versa. On the other hand, our empirical measures are imperfect: both uncertainty sampling and feature ordering by information gain, while relatively efficient, have shortcomings.

Figure 4 illustrates that I_{pc} and F_{pc} of problems computed using SVM uncertainty sampling and LARS are highly correlated ($r = 0.95^4$). The plots of I_{pc} vs. F_{pc} computed using perceptron committees and information gain look similar, albeit with a slightly lower correlation coefficient ($r = 0.81$ ($p < 2.2e^{-16}$)). The SVM methods show higher correlation probably because they have the same underlying SVM learning, and SVM LARS does a better job of feature ordering for the SVM learner than information gain does for perceptron. Additionally n_i and n_f (computed using perceptron committees and information gain) are also strongly correlated ($r = 0.613$ ($p < 2.2e^{-16}$)) and therefore C_i and C_f are also strongly correlated ($r = 0.682$ ($p < 2.2e^{-16}$)).

We also experimented with random sampling for instance selection. The Table below shows the correlation coefficients for I_{pc} and F_{pc} for various combinations of classifiers, instance selection mechanisms and feature selection mechanisms for these 6 corpora.

⁴ r is Pearson's correlation coefficient, and $r=1$ denotes perfect correlation

classifier	Feature Sel.	Instance Sel.	r
SVM	LARS	Active	0.95
SVM	LARS	Random	0.88
Perceptron	Info. Gain	Active	0.81
Perceptron	Info. Gain	Random	0.79

That instance complexity (the minimum number of instances needed to learn a concept) and feature complexity (the minimum number of features needed to learn a concept) are highly correlated may not be surprising since both are probably related to the Kolmogorov complexity⁵ of the learning problem. That our complexity measures exhibit this correlation substantiates our belief in these measures.

5.2 Difficulty of Domains

Corpus	Instance Complexity Measures			Feature Complexity Measures		
	I_{pc}	n_i	C_i	F_{pc}	n_f	C_f
Tech100	0.04 (0.06)	3.24 (2.23)	0.20 (0.33)	0.07 (0.02)	1.89 (1.43)	0.14 (0.14)
WebKB	0.31 (0.13)	8.75 (0.50)	2.72 (1.04)	0.11 (0.04)	4.00 (2.16)	0.51 (0.47)
Reuters-21578	0.35 (0.13)	8.20 (1.03)	2.93 (1.24)	0.12 (0.07)	4.80 (2.04)	0.69 (0.56)
BNC	0.39 (0.16)	7.93 (1.91)	3.34 (1.73)	0.24 (0.11)	11.47 (3.83)	2.97 (1.60)
Enron	0.46 (0.09)	8.33 (0.87)	3.82 (0.94)	0.13 (0.06)	7.67 (4.42)	1.18 (0.70)
20NG	0.48 (0.04)	10.40 (0.68)	5.04 (0.71)	0.23 (0.08)	10.05 (1.39)	2.32 (0.95)
TDT3	0.48 (0.13)	9.30 (1.06)	4.55 (1.53)	0.20 (0.04)	6.50 (1.78)	1.34 (0.53)
Reuters-RCV1	0.53 (0.14)	10.67 (1.84)	5.81 (2.25)	0.23 (0.09)	7.69 (2.04)	1.81 (0.79)
Industry	0.59 (0.12)	10.34 (1.43)	6.20 (1.71)	0.29 (0.09)	5.97 (1.52)	1.77 (0.61)

Table 3. Difficulty measures for different corpora. Higher the value, more complex the problem. Values in brackets indicate std. deviation. The complexity is computed using the perceptron algorithm & uncertainty sampling for instance selection & information gain for feature ordering and selection.

We now benchmark all 9 corpora as easy or difficult for active learning using our complexity measures. Table 3 shows the complexity of different data sets. By all measures the Tech100 data set ranks as the easiest, followed by WebKB and Reuters. BNC, Reuters-RCV1, 20 Newsgroups and the Industry sector corpora are difficult by both our instance complexity and feature complexity measures. This is better illustrated in the chart in Figure 6. This figure reaffirms the high correlation between instance complexity and feature complexity. That most corpora have problems of varying difficulty is demonstrated by the standard deviation of the scores in Table 3. Even though the BNC corpus is small (less than 3k documents) it falls into the difficult end of the spectrum implying that genre classification is more difficult than subject based categorization.

⁵ The complexity of a string is measured by the length of the shortest universal Turing machine program that correctly reproduces the observed data. Remember that K-complexity is only theoretical and cannot be computed.

The ranking of corpora using F_{pc} computed using SVM with LARS and Perceptron with information gain are also near identical as is illustrated by Figure 5 (We only show a subset of the problems to illustrate this, due to the slow running time of LARS). The ranking of individual problems in these two corpora using F_{pc} computed using these two methods also correlate fairly well ($r=0.73$). The F_{pc} scores for individual problems in the Reuters-21578 and 20 Newsgroups corpus using both methods are illustrated in Figure 7. Our results also support previous results that say that 20-Newsgroups consists of problems that are more difficult than Reuters-21578 and that problems like *wheat* are much easier with lower feature complexity as compared to *acq* [2, 13].

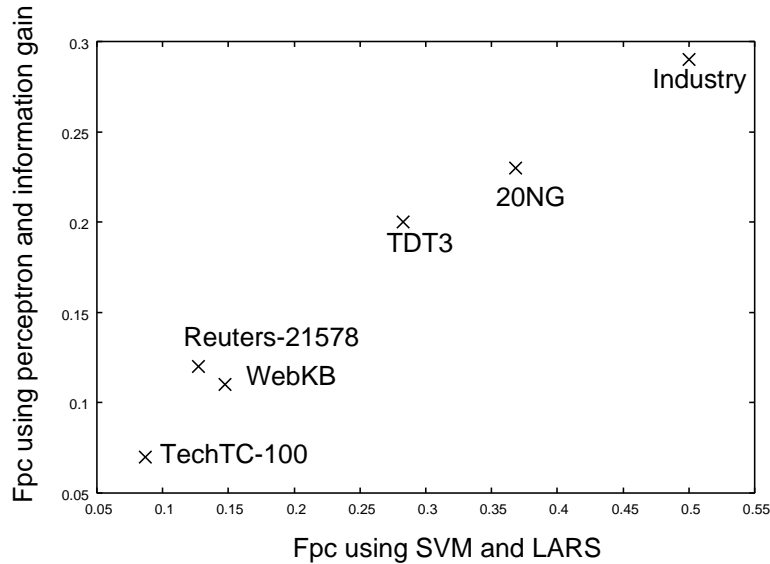


Fig. 5. Ranking using F_{pc} computed by two different methods results in a similar ranking of corpora.

The Tech100 data set is a result of the efforts of Davidov et al [7] to obtain a data set containing problems of varying difficulty in terms of maximum performance achievable. Yet we find all of the problems in this data set are of low complexity i.e., a few well chosen examples or features are sufficient to achieve the optimal accuracy.

The TDT corpus consists of English newswire documents (Eng News), the output of an automatic speech recognizer system for English broadcast sources (Eng ASR), machine translated newswire sources (MT News) and broadcast sources in Mandarin preprocessed through an ASR system and a machine translation system (MT ASR). We measured the difficulty of each of the subsections of this corpus. The C_f values for event based categorization are shown in the second column of Table 4.

The English sub-section of the corpus is easier than the machine translated one, which is more noisy. For example, topic 30036 is *Nobel Prizes Awarded*. The feature complexity of this problem in each subset is shown in the third column. The most im-

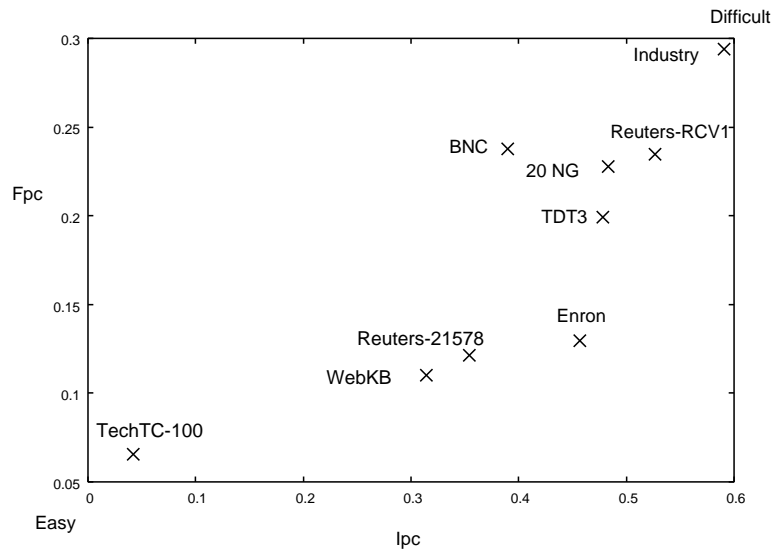


Fig. 6. Instance Complexity (I_{pc}) and Feature Complexity (F_{pc}). A higher value of complexity indicates a difficult problem. Notice how instance complexity and feature complexity are correlated.

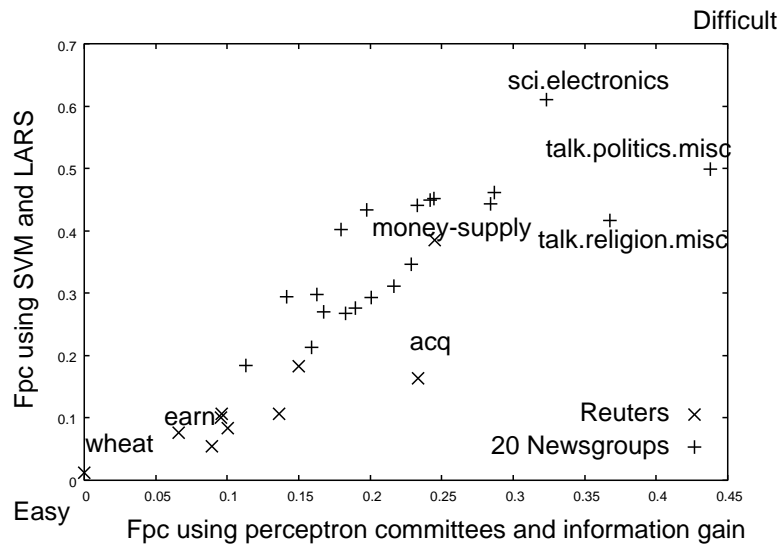


Fig. 7. Feature complexity (F_{pc}) scores of problems in the Reuters-21578 and 20 Newsgroups corpora computed using 2 different methods. Higher the complexity more difficult the problem.

portant words in English Newswire and English ASR are (as expected) *Nobel*, *prize*, *Saramago* (person who won it) etc, making classificaton in Eng-News relatively easy. However, in MT News and MT ASR the most important keywords are *promises*, *Bell*, *prize* and *award*. The word *Nobel* is consistently translated to *promises Bell* in documents whose original source is *Mandarin*⁶. Names like *Saramago* which are highly discriminatory in English are out of vocabulary in the MT documents, making the classification problem even harder. Additionally, a multi-source setting (newswire, broadcast and multiple languages) can be more difficult than considering each source alone as the vocabulary across sources differs depending on the MT and ASR systems used.

Subset of TDT3	C_f by class type			
	Events	<i>Nobel Awarded</i>	Subject	<i>Legal & Criminal cases</i>
Eng News	0.65	0.27	2.03	2.56
Eng ASR	0.95	0.14	2.02	2.78
MT News	1.38	3.25	2.12	2.61
MT ASR	1.22	3.48	1.50	2.03
Whole corpus	1.34	1.60	2.78	3.30

Table 4. Difficulty of the TDT corpus when broken down by source and by category type.

So far we have considered categories based on events in the TDT corpus and *Hurricane Mitch* and *Hurricane George* were different categories. The TDT corpus is also annotated by broader subjects like *natural disasters*, *elections* etc, the feature complexity of which are given in the fourth column of Table 4. The fifth column shows the C_f values for an example topic - *legal and criminal cases*. The important features for classifying by subject are words like *court*, *law* etc., which do not suffer from as many MT and ASR errors making the difficulty of subject based classification about the same in each source type, and even in the whole corpus (see the 4rth column of Table 4).

6 Implications for Human-in-the-Loop Learning

The dual nature of complexity seems to imply that an intelligently picked feature can be as good or better than an intelligently picked instance. That means (in theory at least) that we can actively learn by intelligently picking and weighting features. Of course, labeling features may not be cognitively as easy as labeling instances. A human, with sufficient knowledge of a category, would be able to label almost all the instances (with the possible exception of the relatively few fuzzy instances) with category labels, whereas labeling all the features (with category labels or even merely indicating their relevance) may not be as easy. Relevance of a feature can depend on factors such as the corpus and the learning algorithm used. For example, it is not easy to determine whether the feature

⁶ Nobel is a 3 character word in Mandarin, the first of two of which also correspond to the English word *promises* and the third of which corresponds to the English name *Bell*

drivers is relevant in discriminating between *comp.graphics* and *ms-windows.misc*. Our initial guess was that humans may be able to judge a few features fairly quickly, and that labeling these few features would be equivalent to labeling a handful of documents, but the latter would be more time consuming. Our preliminary experiments showed us that labeling a feature is more than five times faster than labeling an instance. We found that users can pick the most predictive features fairly accurately [22]. For low feature complexity problems, learning may be stopped once features are picked. For medium complexity problems, the user may need to mark a few instances in addition to the features to achieve an acceptable level of accuracy. For very complex problems feature selection may be much more difficult for the user and instance feedback is the more reasonable alternative. Hence, we think a tandem approach of asking on instance feedback and feature feedback is most beneficial: if the problem is of low complexity, a few features that the user marks will quickly lead the classifier to convergence; if the problem is of high complexity, the user would not be able to recommend features (they may not be obvious) but can provide feedback on instances instead. In Section 6.1 we show evidence for this hypothesis.

6.1 Experiments

We saw that instance complexity and feature complexity are two sides of the same coin – a problem for which a few intelligently chosen instances can be used to build a good classifier is also one for which a few good features are very good predictors of class membership. Additionally, we have observed that users can identify the most relevant features with reasonable accuracy [22]. The same work found that labeling features is about 5 times faster than labeling documents. From these results we hypothesize that coupling intelligent feature selection with intelligent document selection should accelerate active learning. Asking users to come up with features apriori is quite difficult. Users found it difficult to determine the relevance of a feature, without having seen any relevant documents. Hence an interleaved approach of asking the users to mark relevant features in tandem with documents that they label is probably cognitively easier. Additionally, in subsequent work we have found that native English speakers could fairly easily point out machine translation errors of the kind discussed earlier where “Nobel” was consistently erroneously translated as “promises bell” [21]. That feedback significantly improved system performance.

In our previous work we built an active learning system for simultaneous document and feature feedback and showed that this dual feedback mechanism results in a much faster learning rate than traditional uncertainty sampling using a support vector machine as described in Section 3 [22]. In our InterActive Feature Selection algorithm, each time a document was picked by uncertainty sampling, the user was also asked to label 10 features. These features were obtained by ranking the features by their information gain scores on the current labeled set, where the labels asked on features were *relevant* (is the feature a discriminatory) or *non-relevant/don't know*. The labeled features were incorporated into the learning by scaling the value of that feature in all the instances. User feature feedback was simulated using an *oracle*, the details of which can be found in our paper. We found that actual users could emulate the oracle to an extent that resulted in as much improvement as can be achieved using the oracle.

The active learning convergence profile p_{al} measures the rate of convergence or the speed of learning. We measured the speed of traditional uncertainty sampling (document feedback only) and that of the InterActive feature selection algorithm for all 358 problems benchmarked in this work. We measure performance only upto $T = 42$ labeled examples and plot the active learning convergence profile (p_{al} , refer Equation 1). Similarly we measure p_{ifs} as the interactive feature selection convergence profile. Figure 8 plots the quantity $p_{ifs} - p_{al}$ for all 358 problems benchmarked in this work. The improvement in speed due to the incorporation of term feedback in addition to document feedback is inversely related to feature complexity as seen in Figure 8 ($r = -0.65$). Speed is improved by about 57% on average.

The *faculty* class in WebKB shows significant improvement in speed (see Figure 8). For this problem, the keywords *faculty* and *professor* are sufficient to obtain 93% of the maximum achievable accuracy (90.05% F1). Both these terms appear for feature feedback within the first 5 iterations in all 30 trials. Similarly, for the Enron corpus, one of the folders is almost completely classified by the sender of the e-mail, *Wilson Shona* (there are some other folders that contain some e-mails by *Wilson Shona*). The algorithm recommends his e-mail id for feedback in the early iterations, resulting in significant improvements in performance. The *miscellaneous* category in the BNC corpus does not gain from term feedback whereas *arts/cultural material* does, because of discriminatory keywords like *opera*, *actor*, *theater* etc in the latter category that when marked relevant improve performance significantly. There are a couple of outliers like the RCV1 category *reserves* for which speed decreases by a large amount when term feedback is included. This may be because a fixed scaling factor of 10 for the selected features is used in the algorithm, which may not be appropriate for every problem. An interesting question is whether there are more robust methods for asking and taking feature feedback into account.

We report the performance (F1) for 8 corpora in Table 2, after 12 and 32 rounds of document feedback using traditional active learning and 12 rounds of interactive feature selection. Interactive feature selection always improves performance over active learning with only 12 documents. It is significantly better than 32 actively sampled documents for 5 of 8 cases. The categories in the BNC corpus are by genre and this result can be interpreted quite intuitively: for categories like “prose” and “poetry” even intuitively it does not seem like there are any keywords that can capture these concepts.

In this section we used our difficulty measures to better understand situations when such methods might work specially well. We have found that feature feedback accelerates active learning by an amount that is inversely related to the feature complexity of the problem. For low to mid range feature complexity problems, a few training documents combined with feature feedback can give a big improvement in accuracy with little labeled data. Many problems in our 9 corpora fall in a low to medium ($0 < C_f < 2$) range of complexity and stand to gain from such a dual feedback framework, automated email foldering being one such domain. Future work includes using these or similar measures to explain other observations, such as when other semi-supervised techniques may work well, as well as exploring methods for predicting the expected difficulty of a learning problem at the beginning stages of training (when few labeled data is available). This can inform the subsequent learning strategy taken.

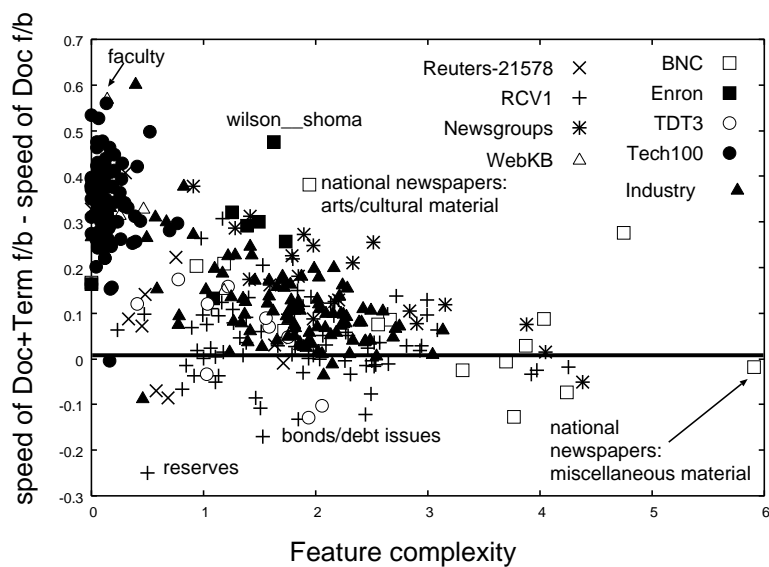


Fig. 8. Difference in speed of active learning and InterActive Feature Selection as a function of complexity (C_f)

Corpus	C_f	Only docs (Active)		IFS
		12 docs	32 docs	
Tech100	0.20	0.486	0.594	0.847
WebKB	0.51	0.262	0.424	0.520
Reuters	0.69	0.516	0.570	0.651
Enron	1.18	0.218	0.444	0.465
TDT3	1.34	0.202	0.259	0.336
Industry	1.77	0.071	0.123	0.199
RCV1	1.81	0.134	0.260	<i>0.231</i>
20 NG	2.32	0.180	0.259	0.336
BNC	2.97	0.209	0.332	<i>0.264</i>

Table 5. Improvement in F1 for corpora of different levels of difficulty. Numbers in bold indicate that InterActive Feature Selection is significantly better than when only documents are used for feedback. Numbers in italics indicate significantly lower performance than the case when $T = 32$.

7 Related Work and Discussion

The classic “curse of dimensionality” informally states that the higher the dimension of the problem, the harder the problem. (in this case, learning). However, our work goes beyond that and tries to measure the the inherent complexity of the problem. A large dimensional learning problem may be easy if only few features are required for learning it. We show here that actively picked examples reveal the complexity better, and we relate this to measures of feature complexity as well.

Note that capturing the exact underlying complexity relates to maximum compression of a given string and is intractable. Thus the subject of this work was to explore the utility of our approximate measures, which depends on the learning algorithm used as well as our chosen instance and feature selection techniques. We reported comparisons in the choice of the learning algorithm and instance selection and feature selection methods (Figure 5 and Figure 7). We have aimed to measure rough complexity, useful for tasks such as comparing and ranking problems. Furthermore, our complexity measure are based on the logarithmic scale, akin to the Richter scale. Intuitively, the same fixed absolute difference, say in the number of instances, is not as important as the number of instances required for learning increases. The logarithmic scale is not as sensitive to relatively small differences in the performance of different instance or feature selection methods.

Ho and Basu [11] defined a set of measures that captured the complexity of the geometry of the boundary for a few artificial and real binary classification problems of low dimensionality. In comparison, our work is in the domain of text classification, where a linear hyperplane is often effective making the geometry of the boundary less of an issue. We experimented with one of their measures of feature complexity -maximum Fisher discriminant ratio, to find that it did not correlate as well with I_{pc} ($r = 0.2$). We also measured how F_{pc} correlated with maximum accuracy and found the correlation to be not very high ($r=0.4$).

For other domains where active learning is used [29] but where the classifier is not linear it is less clear whether our complexity measures can directly be used and we would be interested in exploring this question in the future. The difficulty in domains such as text is that relatively large amounts of training data (100s or thousands) may be needed to converge to the optimal hyperplane. We note that linear classifiers do well on a number of challenging high dimensional real-world problems, for example in natural language and vision [24, 25].

Davidov et al [7] developed a benchmark data set consisting of 100 text-classification problems with varying difficulty (accuracy ranging from 0.6 to 0.92). They also developed measures for predicting the difficulty of a problem, but this was in terms of its accuracy. Instead our focus is in understanding how many features or examples are needed to achieve the maximum accuracy. In fact their data set, Tech-100, is the easiest data set for active learning, and illustrates the fact that difficulty in term of accuracy value is different from difficulty in terms of sample size or feature size requirements.

Gabrilovich et al defined a feature complexity measure *outlier count* [10] that attempts to capture the number of important features for a given learning problem. They used outlier count to characterize problems for which decision trees are more accurate than SVMs, the latter being the main thrust of their work. The work here on the

other hand is an in-depth analysis of complexity – both feature and instance. We did experiment with outlier count finding that it correlates with instance complexity (I_{pc}) reasonably well ($r=0.610$) as our feature complexity measures.

Previous work has noted the “low feature complexity” of problems in commonly used data sets (e.g., [2, 13, 12]). Our work is an attempt to quantify these observations, in particular in high dimensional problems such as text. Blum and Langley [3] provide a good introduction and motivation to this work. They discuss the problem of selecting relevant examples and relevant features as two ways of gathering relevant information in a data set. They formally define the relevance of features and examples, and suggest using relevance as a measure of complexity. Their work is however theoretical and their definitions apply for classes which can be completely described (i.e., 100 % accuracy is achieved) by some conjunction or disjunction of features. Real world problems like text classification are not so simple and it is not clear how their measures may be used to quantify complexity for real world problems. They conclude their paper by stating the following *empirical challenge*:

Feature selection and example selection are tasks that seem to be intimately related and we need more studies designed to help understand and quantify this relationship. Much of the empirical work on example selection has dealt with low dimensional spaces, yet this approach clearly holds even greater potential for domains involving many irrelevant features. Resolving basic issues of this sort promises to keep the field of machine learning occupied for many years to come.

Our measures attempt to address the unsolved questions in their paper. We define measures that can be computed easily in real world domains, and demonstrate that instance complexity and feature complexity are highly positively correlated on the problems we tested.

8 Summary

Designing adequate empirical measures of difficulty is a balancing act between efficiency and utility. The techniques proposed here are simple and efficient, and we presented evidence that they exhibit desired properties in the domain of text classification: rough but useful measures of difficulty, leading to a consistent ranking of problems, and exhibiting explanatory power, for example in explaining the extent of benefit from feature feedback during active learning. We observed a high positive correlation between our instance complexity and feature complexity measures, indicating that they approximate the inherent complexity well. We benchmarked 9 corpora and 358 problems and used these measures to gain insights on the relative difficulty of a variety of text classification problems and domains. Our measures also capture how difficulty can be different even within a corpus depending on the type of classes (say subject or event) that one is trying to learn. We found that problems with low to medium feature complexity stand to benefit most from feature feedback, and we see that many experimental problems, such as email categorization, fall in this range (Table 3 and Figure 8). This

has encouraging implications for the use of active learning with feature feedback for real world filtering and email classification problems.

We hope that our analyses and domain rankings would serve to inform future research, for example in selecting corpora and anticipating results. Future work includes extending these measures and exploring other factors that may help further explain difficulty and variations in learning performance. Candidate factors include the proportion of positive instances and the dependency patterns among the features. It would also be useful to study difficulty measures in other learning problems and domains.

We note that our measures serve primarily for understanding or explaining learning behavior, such as convergence. Currently, they cannot be used to predict complexity on a new problem with no or few labeled instances. Such *prediction* of complexity appears to be a difficult if not an unsolvable task, unless real world problems turn out to satisfy helpful properties. Given that we do not know at the outset how to predict whether a concept is going to be easy or difficult, a tandem learning approach that mixes both feature and instance feedback, may be the best general approach for fast learning, especially in the early stage of learning.

References

1. C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *Information Systems*, 12(3):233–251, 1994.
2. R. Bekkerman, R. El-Yaniv, Y. Winter, and N. Tishby. On feature distributional clustering for text categorization. In *SIGIR 01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 146–153, 2001.
3. A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.
4. J. Brank, M. Grobelnik, N. Milic-Frayling, and D. Mladenic. Feature selection using linear support vector machines. Technical report, Microsoft Research, 2002.
5. N. Cancedda, N. Cesa-Bianchi, A. Conconi, C. Gentile, C. Goutte, Y. Li, J. Renders, and A. Vinokourov. Kernel methods for document filtering. In *TREC 02: The Eleventh Text Retrieval Conference*. Dept. of Commerce, NIST, 2002.
6. S. Dasgupta, A. T. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *COLT*, pages 249–263, 2005.
7. D. Davidov, E. Gabrilovich, and S. Markovitch. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *SIGIR '04*, pages 250–257, 2004.
8. A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 493–500, New York, NY, USA, 2006. ACM Press.
9. J. Feldman. Minimization of boolean complexity in human concept learning. 407:630–633, 2000.
10. E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: Using aggressive feature selection to make svms competitive with c4.5. In *Proceedings of ICML 04: The 21st International Conference on Machine Learning*, pages 321–328, 2004.
11. T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(3):289–300, 2002.
12. R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11(1):63–90, 1993.

13. T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.
14. T. Joachims. Text categorization with support vector machines: learning with many relevant features. In *ECML 98: The 10th European Conference on Machine Learning*, pages 137–142, 1998.
15. S. Keerthi. Generalized LARS as an effective feature selection tool for text classification with SVMs. In *Proceedings of ICML 05: The 22nd International Conference on Machine Learning*, August 2005.
16. D. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML 94: The 11th International Conference on Machine Learning*, pages 148–156, 1994.
17. D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.
18. A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996.
19. N. C. N. Stewart. The effect of category variability in perceptual categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 28:893–907, 2002.
20. F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 23–32, New York, NY, USA, 1999. ACM Press.
21. H. Raghavan and J. Allan. Passage feedback for news tracking. Technical Report IR-459, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, 2006.
22. H. Raghavan, O. Madani, and R. Jones. Interactive feature selection. In *Proceedings of IJCAI 05: The 19th International Joint Conference on Artificial Intelligence*, 2005.
23. F. Rosenblatt. Two theorems of statistical separability in the perceptron. pages 421–456, 1959.
24. D. Roth. Learning in natural language. In *IJCAI*, 1999.
25. D. Roth, M.-H. Yang, and N. Ahuja. Learning to recognize objects. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–731, 2000.
26. R. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge into boosting. In *Proceedings of ICML 02: The 19th International Conference on Machine Learning*, 2002.
27. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 2002.
28. R. Shepard, C. L. Hovland, and H. M. Jenkins. Learning and memorization of classification. 75:1–42, 1961.
29. S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, pages 107–118, New York, NY, USA, 2001. ACM Press.
30. S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66, 2002.
31. V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2 edition.
32. X. Wu and R. Srihari. Incorporating prior knowledge with weighted margin support vector machines. In *Proceedings of KDD 04: Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 326–333, 2004.