TANDEM LEARNING: A LEARNING FRAMEWORK FOR DOCUMENT CATEGORIZATION

A Dissertation Presented

by

HEMA RAGHAVAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2007

Computer Science

TANDEM LEARNING: A LEARNING FRAMEWORK FOR DOCUMENT CATEGORIZATION

A Dissertation Presented

by

HEMA RAGHAVAN

DEDICATION

Dedicated to my father

Rangachari Raghavan

In loving memory of my mother

Saroja Raghavan

ACKNOWLEDGMENTS

Graduate school has been a memorable and fun learning experience in more ways than one and I feel indebted to the presence of several mentors, colleagues, friends and relatives without whom this dissertation would not have been possible.

First and foremost, I would like to thank my advisor Prof. James Allan for his continuous support and guidance over the six years. Prof. Allan is an excellent mentor with a unique combination of technical expertise and human skills. He has always been involved with my work and has provided necessary and valuable feedback (often peppered with wit and humor), has always asked good questions and steered me in the correct direction when necessary. I appreciate his guidance style which provides a considerable degree of freedom to his students, allowing them to pick and chose research problems and techniques of their choice, the result of which is that his students have significant breadth in information retrieval at the end of their PhD. I have learned many valuable lessons in understanding how to tackle a research problem, being open and unbiased about techniques and approaches and the importance of experimental evaluation from him. I hope I will remember these throughout my scientific career. I will also be forever grateful to Prof. Allan for honing my technical writing skills by going through several drafts of my research papers over the years.

I would like to thank Prof. Bruce Croft for his contributions to my work throughout my graduate career. Bruce has an uncanny ability of asking defining questions which when answered correctly help flesh out the arguments for or against a piece of work. He is virtually a library of information on the field of information retrieval due to which he is almost always able to place any piece of work in context with past work. I am grateful to him for going through this dissertation with care and patience.

I would also like to thank Professors David Jensen and Donald Fisher for serving on my committee, for reading through drafts of this thesis and for their valuable insights and comments.

Many of the ideas for this dissertation stemmed from an internship in 2004 at Yahoo! Research under the mentorships of Dr. Omid Madani and Dr. Rosie Jones. Over the years we have maintained our academic relationship through weekly phone meetings. Much of my initial training in machine learning comes from my interactions with them. They both have the curiosity and drive necessary to understand the underlying processes that cause a technique to work. I have thoroughly enjoyed our weekly meetings and the brain storming sessions. They have also been mentors in a broader sense, offering me advice on my career and several inputs on my dissertation.

Being a part of the Center for Intelligent Information Retrieval has its advantages in the variety of students the lab consists of, each of whom specializes in a unique area of information retrieval. The resulting atmosphere in the laboratory and in group meetings is highly energetic with expertise in a variety of topics. Although each student tackles their research independently there is ample discussion and collaboration, contributing to a unique learning experience. I am particularly grateful to Ramesh Nallapati, Vanessa Murdock, Fernando Diaz, Jiwoon Jeon, Ron Bekkerman, Giridhar Kumaran, Xiaoyong Liu, Mark Smucker, Xing Wei, Trevor Strohman, Ben Carterette, Donald Metzler, Xing Yi and Niranjan Balasubramanian for valuable discussion. David Fisher must be especially thanked for all the feedback he has offered on written drafts of papers and critiques of my research talks. I am also grateful for the opportunity to have worked with Prof. Andrew McCallum during my graduate career.

I am also thankful to all those who make the computer science department run like a well oiled machine: Kate Moruzzi, Sharon Mallory, Pauline Hollister, Jean Joyce and Andre Gauthier deserve a special mention in this regard. The brisk efficiency and warm smiles with which they execute tasks allow graduate students to focus on courses and research making administrative bottlenecks seem almost non-existent.

Graduate school, while fun for the most part, has its challenges. There are days of grilling course-work, paper deadlines, rejected papers, qualifiers, and probably the most dreary of them all - long periods when “nothing works”. Those days would have been impossible to get through without the support of family and friends. The endless cups of “chai” with friends, the mindless jokes, the heated debates, long lunches solving Daily Collegian crosswords, even longer dinners, summer evenings in Smiarowski farms, the long fall drives and impromptu hikes in the neighborhood, more than made up for the rough days.

I have to thank the large circle of friends I’ve had at UMass and outside each of whom has been special and dear. I feel particularly thankful for having had excellent housemates over the years, especially Kausalya Murthy who has been a great support and a loyal friend throughout the years. Our house at 55 Presidential has become synonymous with home for me, providing a safe haven when I return home from a long day in the lab. Swati Birla deserves a special mention as being a close friend and a continuous source of support, always lending an nonjudgmental ear through good times and bad. I cannot imagine how lab would have been without Ramesh Nallapati, who at this point is a sounding board for every idea and thought in my head whether academic or non-academic. His infectious laughter, warmth and affection have made my stay in Amherst a more than pleasant experience. Sudarshan Vasudevan with his level-headed sensibility, relaxing company, warmth and an ability to make me laugh in times when I have thought myself incapable of doing so, has also been a companion I cannot imagine what Amherst would have been like without. Vanessa Murdock has also been a close friend as well as a valued colleague. In my initial years she helped me understand and assimilate American culture. Till date our backpacking trip to the north of Finland after SIGIR 2002 has been one of my best outdoor trips. Purushottam Kulkarni who kept encouraging me to “just send it” in his own quiet way, Nasreen Abdul-Jaleel with her sheer energy and passion for life, and Bruno Ribeiro with his good cheer, affection and the tact to narrate the most banal episode with rib-tickling humor made the days at Amherst brighter. Sharad Jaiswal was one of my first few friends at Amherst and through him I learned to know and grow to love the valley. The Patwardhans (Tanmay, Shanta aunty and Sudhir uncle) have been a source of encouragement, help and support in my life for many years now. Likewise Kanta Jiwnani, my friend from high school, has shown me affection and care for so many years now, that I wouldn't know where to begin thanking her. I must also thank my other friends in Amherst – Ashwin Gambhir, Tejal Kanitkar, Preyasee Kamath, Abhishek Chandra, Pritesh Sharma, Giridhar Kumaran, Ashmita Sengupta, Pallika Kanani, Ravisubhash Tangirala, Aparna Vemuri, Bhuvan Urgaonkar and Emily Horrell for their friendship and support. Their constant companionship will be greatly missed.

It also helped that many of my friends were computer science graduate students and we were able to identify with each others experiences of graduate school. I will particularly treasure the discussions with Ramesh and Sudarshan on what constitutes good science and I hope we will all adhere to the standards of quality in our work that we thought were imperative.

I must also thank my cousins Vidya Madhavan and Jagadeesh Yedetore for providing a weekend home in Boston and pampering me when I went there. Jaggi's gourmet cooking was always more than a pleasant break from the hurriedly prepared dinners that characterize graduate school life. My niece and nephew Aditya and Yamini offered playful respite and I often forgot the tensions of graduate school in their company.

My aunt Jaya and my cousin Radhika and their relentless spirit have been a source of inspiration for me throughout my life. In any moment when my spirit is low, I look to their lives as sources of strength, courage, and optimism for myself. I thank my nieces Soumya and Preeti Iyengar for all the love and affection they shower on me.

I have grown to realize that this point in one’s life, where a dissertation is complete and graduate school is past, is not a journey that began with admission to graduate school. Rather many of the foundations are laid earlier in one’s life, and for this I owe much credit to my parents. They instilled in me their belief in a sound education and a pursuit for knowledge ever since I can remember. My mother always had the desire to learn. She would find a topic, master it and apply her new found knowledge incredibly well. Every new milestone only wanted her to aim higher. Her example also taught me that if you set your mind to a task you can accomplish it. She was a strong-willed, independent and highly motivated person and I will be happy if I have imbibed even a fraction of her qualities. In my father, I found a model of honesty and integrity, from whom I learned to pursue science for science’s sake and for the sake of curiosity. Both my parents have been incredibly strong and patient people and highly dedicated as parents who have made many sacrifices to ensure that I have a good life and a successful career. I owe everything I am to them. I hope that in my dissertation and in my future life I can do them justice.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by SPAWARSYSCEN-SD grant number N66001-02-1 and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsor.

ABSTRACT

TANDEM LEARNING: A LEARNING FRAMEWORK FOR DOCUMENT CATEGORIZATION

MAY 2007

HEMA RAGHAVAN

B.E., VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE, MUMBAI

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor James Allan

Supervised machine learning techniques rely on the availability of ample training data in the form of labeled instances. However, in text, users can have a strong intuition about the relevance of features, that is, words that are indicative of a topic. In this work we show that user prior knowledge on features is useful for text classification, a domain with many irrelevant and redundant features. The benefit of feature selection is more pronounced when the objective is to learn a classifier with as few training examples as possible. We will demonstrate the role of feature feedback in training a classifier to suitable performance quickly. We find that aggressive feature feedback is necessary to focus the classifier during the early stages of active learning by mitigating the Hughes phenomenon. We will describe an algorithm for tandem learning that begins with a couple of labeled instances, and then at each iteration recommends features and instances for a user to label. The algorithm contains methods to incorporate feature feedback into Support Vector Machines. We design an oracle to estimate an upper bound on tandem learning performance. Tandem learning using an oracle results in much better performance than learning on only features or only instances. We find that humans can emulate the oracle to an extent that results in performance (accuracy) comparable to that of the oracle. Our unique experimental design helps factor out system error from human error, leading to a better understanding of when and why interactive feature selection works from a user perspective. We also design a set of difficulty measures that capture the inherent instance and feature complexity of a problem. We verify the robustness of our measures by showing how instance and feature complexity are highly correlated. Our complexity measures serve as a tool to understand when tandem learning is beneficial for text classification.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

b	The threshold of a linear classifier of the form $f(X) = w \cdot X + b$.
B_f	Allowance or budget of the number features permitted to be asked in a given run of active learning.
C	User specified constant to an SVM training algorithm, specifying the maximum extent of influence of a support vector.
E_T	Efficiency of learning, measured after T examples have been actively picked.
f	Number of features presented to the user in a tandem learning iteration.
$\mathcal{F}$	$\mathcal{F} = f_1 ... f_k$, indices of features presented to the user during an iteration of tandem learning.
$F1$	Measure of effectiveness. Harmonic mean of P and R.
$F1_t(ACT)$	F1 of a classifier trained on t actively picked examples. $F1_t(RAND)$ is similar, but where examples are randomly picked.
$F1_t(ACT, k)$	F1 of a classifier trained on t actively picked examples and with k features $F1_t(RAND, k)$ is similar but where examples are randomly picked.
I	Number of instances presented in a given active learning iteration.
k	Number of features picked by the user in a tandem learning iteration.
$\mathcal{M}$	The classifier or the model. In our case typically $\mathcal{M} = (w, b)$.

M	Size of the pool.
N	Number of features.
P	Precision of a classifier.
R	Recall of a classifier.
t	Number of training examples.
T	Total number of training examples.
$\mathcal{U}$	The set of unlabeled examples an active learner can query from.
w	$w = w_1 ... w_N$. A vector of weights on features in a linear classifier of the form $f(X) = w \cdot X + b$.
X_i	An instance. Also called an example or a document.$X_i = x_{i1} .... x_{iN}$.
Y_i	Label of an instance i.
λ_i	Extent of influence of a support vector.
ξ_i	Error tolerated for a misclassified support vector.

CHAPTER 1
INTRODUCTION

A fundamental task in human reasoning is the grouping or clustering of similar objects together and drawing generalizations about them. Categorization involves associating a semantic label to a group of similar objects. The idea that categorization is fundamental to knowledge and reasoning goes as far back as Aristotle [8]. Concepts are representations of categories and many inferences that we draw about our surroundings – from “a bird is an animal” to an “atom is the smallest particle of matter” – can be thought of as associating categories to observations. How concepts are represented in the human brain is an interdisciplinary field spanning brain and cognitive sciences to ontology and metaphysics. Many theories exist with little consensus [84].

Given the growing volume of electronic information on the internet and in digital libraries, one would like computers to be able to automatically organize documents into groups that are intuitively perceptible to humans like topic, genre, sentiment and so on. This organization can be achieved in one of two possible ways: through unsupervised clustering algorithms or supervised categorization algorithms. The task of grouping documents by their similarity together, with no human supervision or information specifying which kinds of documents should go together is typically called **clustering**. The task of assigning category labels to documents is called **categorization** or **classification**. Labels may be based on topics e.g., *earthquakes*, genre e.g., *poetry*, geography e.g., *china* and so on. Categorization is typically a *supervised* task with a human specifying the categories of interest and some additional information that is used to generate a general rule that can map a previously unseen document to one of the pre-specified categories.

Document clustering as an effective means of storage and retrieval of information has been studied since the 1960s [28, 86, 80, 95]. Traditionally the purpose of such a grouping in an information retrieval system was indexing, where each document was assigned one or more key-words or phrases describing its content. The keywords were typically part of a controlled dictionary. Each of these keywords can be viewed to be a category and the problem of associating a keyword to a document can be considered analogous to associating a class label to it. van Rijsbergen's classic book on information retrieval [111] has additional references to some of the early work in automatic text classification and clustering. An informal note prepared by Sparck Jones also surveys much of the early literature in this area both in information retrieval and out of it [103]. As she points out, much of the statistical literature available in the 1970's was not directly applicable to the task of retrieval. The field therefore evolved into designing its own techniques for grouping documents. On reading some of these early works, the reader will notice that the term classification was often used to refer to the clustering task as defined above. However in keeping with the current literature we use the term classification to exclusively mean categorization in our work.

The AIR/X system built for a large physics database represents one of the biggest efforts in information retrieval to build a classification system for indexing using a controlled vocabulary. Although the main purpose was indexing, the authors have on occasion evaluated their system on a document classification task [45]. The initial systems were heavily rule based [45], but over the years several statistical techniques were developed and evaluated for the indexing task [65, 110, 18].

Research in the task of text categorization gained momentum in the 1990s with the development of several statistical methods. A survey published by Sebastiani in 2002 [98] provides a very good overview of the techniques in automatic text categorization. The field today lies at the cross-roads of information retrieval and machine learning. Machine learning is an area of computer science that concerns itself with techniques that enable computers to "learn", allowing computers to automatically do tasks that humans can. Modern day text categorization algorithm fall into the inductive learning paradigm of machine learning. Inductive learning algorithms typically take as input sets of example documents in the categories of interest and automatically learn a function that can accurately map a new unseen document to one of the many pre-specified categories. Machine learning algorithms, in striving to solve a more general task, typically limit the kind of input that may be obtained from a human to labels on training examples. They therefore overcome the primary disadvantage of older rule-based systems that typically did not generalize well across corpora. Machine learning techniques however aim to be generalizable across domains as well i.e., strive to build one classification algorithm that works well for text classification, gene classification, image classification and so on. In doing so, they lose out on the benefits that may be obtained by considering the domain. For some application domains there may be other types of inputs that may be more easily procured, other than labels on training examples. For example, it may be quite easy for a human to specify some simple "soft-rules" in building a text classifier (and doing so may be much easier than designing an entire rule based system).

We argue that if the goal is a task, then we should accordingly take a "task oriented" approach, as has been adopted by the information retrieval community. Otherwise we may lose out on easily available information that is valuable to the learner (classification algorithm). An important problem for both the human teacher and the learner in machine learning is the efficiency or speed of the learning process. We will show that alternate forms of human computer interaction (other than labels on examples) that exploit the domain of the underlying task, can accelerate learning.

Additionally, traditional inductive learning techniques encounter a difficult feature selection problem in the domain of text because of their sheer number: features of a text document are often the words that occur in it and often a collection of a even few 1000 documents can have over 10^4 unique words. We will show that a human can help overcome the feature selection problem by using their prior knowledge to label relevant features. A machine learning algorithm can then use its traditional inference mechanism in conjunction with the knowledge that the human has provided. Information retrieval has studied various kinds of information that a human (naive users as well as domain experts) can specify to a computer in the domain of text. We elaborate further on these techniques in Chapter 3. Text categorization being the focus of this thesis, we argue for an amalgamated approach that takes lessons from approaches of the information retrieval community and that uses the new techniques from machine learning for text categorization. We will show that by bringing in the focus of the task, we can make machine learning algorithms for text classification perform even better than the current state of the art techniques. The new framework, *tandem learning*, aims at learning categories in text faster, that is, with less training effort from a human.

We will show that a combined approach that exploits machine learning techniques and methods for feedback as have been done in information retrieval will be most beneficial for text categorization. In subsequent sections we will explain our ideas and terminology in greater detail.

1.1 Terminology: Analogies from Human Learning

We begin by discussing learning in humans, firstly because machine learning and human learning have many parallels, making it easy to introduce terminology, and secondly because it allows us to use examples developed in this section to more easily present our ideas to enhance the standard mode of human computer interaction in machine learning.

One view of the representation of a concept in the human brain is a “definitional one” where the concept is represented as a definition (or a rule) composed of a set of properties of the category [9]. We use the terms **concept** and category interchangeably. An object or an entity is composed of *features*. The properties of a category or a class are features that are likely to occur in an object belonging to that category. A complete definition filters all objects in the world belonging to a category leaving out all objects that do not. In the case of a bird, its features are its eyes, feet, feathers, beak and so on and its properties would consist of features likely to occur in birds (beak, feathers etc). If all or at least most of the important properties are satisfied, an object would be considered a bird. In the real world most definitions are not complete and even for scientific definitions one can find exceptions. For example, an ostrich is an atypical bird since it is flightless.

Many categories are hard to define by a list of properties, for example the concept “art” is probably not easy to define. The “exemplar view” by contrast, takes a different perspective and assumes that the brain stores all previously observed examples of objects belonging to a category. When a new, unseen object comes along, it is compared to all stored examples, and one infers the category label according to the labels of examples it is most similar to.

How concepts are learned is another interesting question. Learning may be *unsupervised*, where the human learner draws inferences from observations, but much of learning is *supervised* or guided by a *teacher* who tries to explain her representation of a concept to a learner. Supervised learning is especially true for the learning of category *labels*, that is learning that a given set of similar objects all belong to a given category (say birds). In trying to teach (or train) the learner, a teacher often tries to define the concept if possible, and then uses examples leaving the student to develop her own representation of the category. The number of examples required would depend on the concept being learned. For example, learning the concept “art” would be difficult without many examples. Examples may be intelligently chosen by the teacher so as to include a mix of typical and atypical examples. Much of human learning, and especially in the early stages when a new concept is being taught to the learner, is interactive, with the teacher answering questions that the learner is uncertain about.

1.2 Machine Learning: The Exemplar Approach and its Limitations

There are many parallels to human learning and machine learning and although the two fields draw from each other [106], there exist many diverse machine learning algorithms all of which are effective at capturing concepts and it is hard to say whether or not they are emulating the human brain. However, one commonality between them is that like human learning, learning algorithms fall into two broad categories – unsupervised and supervised. Unsupervised learning techniques try to learn inferences about examples without associating categories to them. Clustering objects by their similarity is an unsupervised extreme of the grouping of documents. Associating category labels to instances on the other hand is a supervised learning problem, which is the focus of this work.

Most supervised learning algorithms are "exemplar" in nature. The learner here is a program that takes several examples (and possibly counterexamples) of objects in a category labeled by a human, and uses this input to learn some abstract representation of the category. Naive Bayes, support vector machines (SVMs) and decision trees are examples of such algorithms and each of them infers properties of a category in their own way. By contrast, algorithms like K-nearest neighbors (KNN) [41] memorize all the input examples, and when a new unseen example is considered, it is compared to all the memorized examples, and is assigned a category corresponding to the most common category of the K most similar examples. For many domains KNN may or may not be as effective as the other approaches [116, 57]. In spite of the difference in the inference mechanism, all these approaches, be it SVMs or KNN, take labeled examples as input, a reasonable approach for trying to find a general method for the learning of categories, where the objects to be categorized may be as general as images, genes or documents. The formalism of the problem becomes universal if approached this way: given pairs of examples, with their expected labels, design an algorithm that learns the hidden function that maps an unlabeled example to a label. This formalism however ignores the fact that for many categorization problems (and depending on the domain), we can often solicit some more information from the teacher – their concept definition for example – in order to train the learner. In the next section we discuss what kind of information we can obtain from the human and argue why that may be useful.

1.3 Towards faster learning

The teacher in machine learning, is a human, often the engineer or the user of the system. In the classical exemplar approach, the learner often needs to see many examples on a category before arriving at a concept that generalizes well, that is, classifies unseen examples with adequate accuracy. Usually a large number of randomly selected examples are labeled by the teacher that are then used by the learner to learn a concept.

Labeling examples in this manner can be tedious and costly for the teacher and many methods have been considered to reduce the total number of labeled examples. One such method is active learning [5]. *Active learning* is an interactive, iterative, human-in-the-loop learning paradigm in which a teacher (a human) begins by providing a few typical examples of a category. The learner (a program in this case) learns some initial concept, and then can *query* the teacher on the category of any example that it does not know the category of. The queries are chosen selectively by the learner such that the knowledge of the category of the queried example would provide maximum value to the learner in improving its current representation of the concept. By selectively picking only informative examples as queries, active learning methods strive to decrease the total effort for the teacher. Active learning is analogous to typical classroom learning, where the teacher imparts some information to the students, and then leaves some time for questions. An efficient student will pick the questions that she is less likely to figure out by herself, for the teacher to answer in the limited time frame. Likewise, an efficient teacher will try to optimize the information imparted during the lecture in order to maximize the information gained for the learner, and decrease her effort in the question-answering session.

As with general machine learning methods, typical active learning algorithms are also example based. By contrast, we saw that in human learning the teacher might try to start out by “defining” a category and then giving examples of the same. One can only imagine how long it would take if humans used only the example based approach for learning. Say for example in learning the concept “star”, one can imagine what a time consuming process learning would be, if each individual had to infer all the properties of a star (gaseous, source of light etc) using only examples. Fortunately, in human learning a teacher often imparts her idea of the concept, often accompanied by illustrative examples, leaving the student’s brain to process these two types of information in some way. The question then is one of using this kind of an approach for machine learning. Intuition says that such type of learning should be faster. Often defining the concept may not be a laborious task for the teacher, and may be much less effort than providing several training examples belonging to the category. The definition may be quite sufficient in many cases (like learning the concept “bird”), but many examples may be needed in other cases (like learning “art”). In other cases, like learning the concept “star”, the definition accompanied by a few examples may be sufficient. Given this initial information, the active learner may then proceed by intelligently querying the teacher on the labels of examples. One can also extend the active learning framework to include questions about properties of the category as well; analogous to a real life situation where a learner may ask “is the laying of eggs a property of birds?”.

Of course, the effectiveness of this idea will depend on the concept itself, the domain, the algorithm, the kind of representation it uses for examples and the final concept, and whether a definition provided by a human can be incorporated into that representation. In some applications these aspects of a problem may be less obvious. For example, in classifying images by texture it is less clear whether a human can specify any information other than labels on examples. Whereas, for an algorithm that determines whether an image contains the picture of a *sunset*, specifying the property that the image is predominantly orange may be quite easy for a human. If color histograms are used by the algorithm in the underlying representation then imbibing this information may be easier for the learner. If not, it is less clear how the learner will absorb this information provided by the human into its representation of the concept. Then again, if the process of absorbing this information is noisy, learning may not be accelerated in spite of this intuitively useful information provided by the user.

In the next two sections we will present why we believe that soliciting “definitional” information will be useful for accelerating the process of learning for text categorization in particular. The questions raised through the aforementioned examples also provide the intuition for the set of hypotheses that structure this thesis (Section 1.5).

1.4 Text Categorization and the Motivation for Tandem Learning

In this thesis we will focus on learning categories in text. Automatic text categorization has applications not only in the organization and retrieval of collections of documents, as in the Yahoo! directories[1] or MESH categories [77] but is also useful for news filtering and email foldering. The scenario for a typical news filtering task [72, 3] is one where a user browsing the daily news comes across a story on a topic of interest, say for example, the first story predicting the arrival of *Hurricane Mitch* and decides that she wants to track it. As new stories arrive in the news, if they belong to the category *Hurricane Mitch* they are delivered to this user. The email foldering task [63] is similar: email is automatically organized into user-specified folders as it arrives. Spam filtering [27] is yet another application of text categorization. Information retrieval can also be articulated as a categorization problem [111, 89] where documents satisfying a user’s information need are associated with a *relevant* category, and those that do not are associated with a *non-relevant* category.

To obtain a classifier that generalizes well, the learning algorithm needs to see a large number of labeled training examples. Labeling data is a tedious and costly process and we

[1]`http://dir.yahoo.com`

seek algorithms that learn a good classifier with as little training data as possible. Companies like Yahoo! often employ paid editors to label large amounts of training data. Decreasing training data serves to decrease their editorial costs. In news filtering and email foldering where the user (teacher) must provide training examples (sample relevant and non-relevant documents) in order to train the system, the user is probably willing to engage in very little interaction in order to train the system.

wheat & *farm*	$\rightarrow$	WHEAT
wheat & *commodity*	$\rightarrow$	WHEAT
bushels & *export*	$\rightarrow$	WHEAT
wheat & *agriculture*	$\rightarrow$	WHEAT
wheat & *tonnes*	$\rightarrow$	WHEAT
wheat & *farm*&¬*soft*	$\rightarrow$	WHEAT

Table 1.1. Induced rule set using the CONSTRUE system [7] for categorizing *wheat* documents in the Reuters data set. The induced rules result in 99% accuracy.

Towards this goal of decreasing the training data in text categorization tasks, we believe that the teacher (user/paid editor) may often be willing to impart her idea of the concept to the learner by describing properties of the category to be learned. Features in a text document are typically words/terms that occur in it and most standard text classification algorithms use a bag-of-words (BOW) representation. Correspondingly a concept may then be defined by a set of high probability terms that occur in documents belonging to the corresponding category. Some categories may be sufficiently accurately described by a conjunction or disjunction of words that appear in the text, for example, the *earnings* category in Reuters news articles is probably reasonably accurately described by the property that the words *dollar, cents, revenue, qtr* etc. occur with high probability in them. An example of a set of (automatically induced) rules that filter documents on the category *wheat* in the popular Reuters 21578 corpus are shown in Table 1.1. Although providing such a terse boolean expression (/definition/rule) may require substantial effort, we think that a human can easily point out that the terms *wheat, farm, commodity* etc are discriminatory.

Such information may be easy (cognitively) and quick for a teacher to provide, resulting in significant improvements in performance. If so, then there is a strong association between what information a human can provide and the underlying document representation. We also think that human assisted feature selection will not be sufficient in itself for the same reason that categories and their associated concepts can often not be "defined" concretely by a set of attributes. Many concepts can be subtle: for example, it is not clear what the features constituting an all encompassing definition to filter documents on the category *arts*.

We therefore define tandem learning, an interactive learning paradigm that builds on active learning. In this framework the learner not only asks the user for labels on examples with an aim to decrease the total effort for the teacher, but also has a mechanism to incorporate the teacher's knowledge on the relevance of features and properties of the category. In fact, in our framework, the learner also queries the teacher on features. For example, the learner may ask if the word *winds* was more likely to occur in documents in Hurricane Mitch. We think that leveraging a teacher's prior knowledge on features should accelerate learning over pure example based active learning.

1.5 Tandem Learning: Proposed Idea and Hypothesis

A tandem learning system is an active learning system where the system intelligently picks features and examples for the teacher to label. The learner starts with some initial concept that is learned by a few labeled examples or features, and then at each iteration the learner can query the teacher on the label of an example (like in traditional active learning) or on the property of a category. The question on the property of a category is specifically restricted to one of asking on the relevance or usefulness of a feature for determining whether an object belongs to a category or not. We use the phrase "feature feedback" to imply human assisted feature selection, because in our final algorithm, the human gives feedback on features that the system selects and asks specific questions about.

Rather than formulating the hypothesis as "feature feedback by humans can accelerate learning", we factor the hypothesis into three parts:

1. *That there exist a set of features, for which if the learner is provided relevance information, the speed of active learning can be significantly improved.*

2. *That the learner can pick these features to ask the teacher for feedback on, in a tandem learning like framework.* In the first part we determine that there are some features, such that, if the learner knew they were relevant early in the learning, that knowledge would bootstrap active learning significantly. In the second step we determine if we can come up with an algorithm where the learner can actually ask for feedback on these important features.

3. *And that these features can be marked by humans fairly easily.* Once we show that the learner can pick the necessary features to ask the teacher about, we show that humans can label these features sufficiently well, that is, well enough to accelerate the speed of learning.

Such a factoring out is unique to this work, and we think that it helps separate algorithmic error from human error. We think that such a reasoning is especially critical for researchers in Human Computer Interaction, where often the algorithm is treated as a black-box [62] and it has been difficult to pin point the source of error when a particular type of feedback does not work (Section 3.1).

1.6 Scope of the Thesis

We restrict ourselves to text classification laying out a solid experimental framework for analysis that can be repeated for other domains (discussed in Chapter 9). We also specifically consider only two forms of input from the teacher: information on the labels of examples, and information about whether a feature helps discriminate an object belonging to the category from one that does not (this discriminatory property of a feature is also called the relevance of a feature for a given categorization problem).

In evaluating our methods we consider effectiveness (accuracy of categorization) and the speed of learning by using evaluation measures that incorporate a "learning rate" or more simply by considering effectiveness after limited feedback from the teacher.

1.7 Outline of the thesis

In the next chapter we formulate the problem more formally outlining the proposed algorithm. We then review past work in machine learning and information retrieval in Chapter 3. We also highlight our contributions with reference to previous work. In Chapter 4 we review the materials and methods used in this thesis. A large part of our work, especially the tandem learning algorithms, builds on support vector machines (SVMs) which we describe in detail in that chapter. We also describe active learning using SVMs, feature selection, data-sets, and evaluation measures in that chapter. Many of our interactive experiments rely on an oracle, which gives a sense of the improvements possible using tandem learning. We describe the oracle in Chapter 4 as well.

Chapter 5 proves the first of the three hypotheses. We also try to understand why feature selection helps the underlying machine learning algorithm in that chapter. Using lessons learned from Chapter 5, we then move on to design a tandem learning algorithm in Chapter 6, filling in the gaps in the skeleton algorithm developed in Chapter 2. Results on experiments using the tandem learning algorithm are discussed in Chapter 7. Our experiments were performed using both an oracle and a real user. The success of both these experiments proves hypotheses 2 and 3.

We then define a set of complexity measures that aim to capture how "definable" a concept is in Chapter 8. Our complexity measures serve as a tool to determine the range of problems that exist in text categorization, i.e., how many concepts are definable (like "birds") and how many are not (like "arts"). For example, certain e-mail folders may be completely defined by a few properties like the sender and the recipient of the mail whereas a genre may not be so easy to define. Similarly categorizing Reuters *wheat* articles can also be typically achieved by a few features (Table 1.1). We obtain many valuable insights for understanding when and why teacher feature feedback is useful for text categorization.

Chapter 9 summarizes the lessons learned along the way, laying out directions for future work.

1.8 Contributions of the thesis

- We offer several statistical intuitions and insights explaining why tandem learning is beneficial for text categorization (Chapter 5).

- We design a tandem learning algorithm that extends the active learning approach to not only query the user on labels of examples, but to also query the user on the relevance of features for a task. The algorithm incorporates this feedback in a way that results in significant improvements in performance with much less effort for the teacher (Chapter 6 and 7).

- Our unique experimental design helps separate algorithmic error from human error, bringing out a novel approach for the design of experiments in the field of human computer interaction.

- We quantify the nature of concepts that exist in text (Chapter 8) providing further explanations for why tandem learning should work.

CHAPTER 2

THE TANDEM LEARNING ALGORITHM: OUTLINE

Active learning techniques are sequential learning methods that are designed to reduce manual training costs in achieving adequate learning performance. Active learning methods reduce costs by requesting feedback selectively and intelligently from a *teacher*. The teacher is a human in the text categorization domain. The teacher may also be called the *user*, especially when the teacher training the model is the same as the person using it, for example a user who is training a personalized news filtering system. Traditionally in active learning the teacher is asked *membership queries* which are questions on the class labels or categories of selected instances (documents in our case).

In the first chapter we motivated the need for a machine learning algorithm for text classification that actively learns by querying a human on instances and features in tandem. We argued that such an approach should accelerate the learning process, that is, enable the learner to construct a classifier with less training effort for the human. In the proposed *tandem learning* framework, the teacher is asked questions about the relevance of features in addition to membership queries. In this chapter we formally lay out the proposed tandem learning system, a system that builds on active learning and introduce terminology used in this thesis more formally. We describe the traditional active learning system in the next section and then describe how tandem learning builds on active learning in Section 2.2. We also describe the *oracle*, which plays an important role in our experimental evaluation, in that section.

Standard Active Learning

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), init_size (number of random feedback iterations), I (Number of instances labeled in each round)
Output: $\mathcal{M}_T$ (Model)

```
$t = 1; \mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a. $\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_0, \mathcal{U}_{t-1},\ random)$
   b. Teacher assigns label $Y_t$ to $X_t$
   c. $t++$
2. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t-1\}$, $\mathcal{M}_0$)
3. While $t \leq T$
   a. $\langle X_t, ..X_{t+I-1}, \mathcal{U}_{t+I-1} \rangle$ = Instance_Selection($\mathcal{M}_t, \mathcal{U}_{t-1}, uncertain)$
   b. Teacher assigns label $Y_t..Y_{t+I-1}$ to $X_t...X_{t+I-1}$
   c. $\mathcal{M}_{t+I}$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t+I-1\}$, $\mathcal{M}_t$)
   d. $t = t + I$
Return $\mathcal{M}_T$
```

Figure 2.1. Algorithm and block diagram for traditional active learning where the system asks for feedback on instances only (**System 1**).

2.1 Active Learning

A typical algorithm for active learning is shown in Figure 2.1. An *instance* X_i (which is a document in our case) belongs to a *class* Y_i ($1 \leq i \leq M$, where M is the maximum number of instances available, typically the size of the data set). X_i is represented as a vector $x_{i1}...x_{iN}$ of *features,* where N is the total number of features. Features in text are typically words, bi-grams (adjacent pairs of words) and tri-grams (adjacent triples of words). This representation, more commonly known as a bag-of-words representation, has consistently been found to work well for topic classification [98] [1]. The value of x_{ij} is the number of occurrences of term j in document X_i. We work on binary *one-versus-rest* classification with Y_i taking values of +1 or -1 depending on whether the instance belongs to the category of interest, or not. An instance in the document collection is *unlabeled* if the algorithm does not know its *label* (Y value). The active learner may have access to all or a subset of the unlabeled instances. This subset is called the *pool* (denoted by $\mathcal{U}$).

The algorithm begins by randomly picking $init\_size$ number of labeled instances (Step 1) by calling the *Instance_Selection* subroutine and passing the parameter *random* to it. The model $\mathcal{M}$ is then trained on these randomly sampled instances (Step 2). The subscript on $\mathcal{M}, \mathcal{U}$, X and Y corresponds to the number of instances that have been labeled. Sometimes one may use keyword based search or some other procedure in place of random sampling to obtain this initial set, especially in cases when the proportion of examples in the two classes is highly unbalanced. Next, active learning begins (Step 3). In each iteration of active learning the learner selects a set of I instances from $\mathcal{U}$ using some criterion (e.g., a measure of informativeness) and asks the teacher to label it (step 3.a). In a popular active learning method, called uncertainty sampling, the classifier selects the most *uncertain* instance [75] for a given model ($\mathcal{M}$) and a pool of unlabeled instances ($\mathcal{U}$). The newly labeled instance is added to the set of labeled instances and the classifier is retrained (step

[1] In fact it has been found that even humans can classify documents quite well using bag-of-word representations [42].

3.c). The teacher is queried a total of T times. The *train_classifier* subroutine uses the labeled data as training, as well as the model ($\mathcal{M}$) learned in a previous iteration, allowing for the case of incremental training [39] or the case when the model may be initialized by prior knowledge [115] (also see Section 5.2).

We use a semi-batch approach of active learning, where the teacher is queried on I instances in each round of active learning (Step 3.a to 3.c in Figure 2.2) rather than one. A larger value of I implies a greater (factor of I) savings in time, at some cost to effectiveness because the model $\mathcal{M}$ will not be updated after each instance is selected, and the Instance_Selection subroutine will not be able to exploit the label of a previously labeled example X_{t+i} in choosing the subsequent instance X_{t+j} ($t \leq i < j \leq t + I$). For example, if the learner is using uncertainty sampling (Section 4.1.5), the knowledge of the label of X_{t+i} may reduce the learner's uncertainty of the label of X_{t+j}, making querying on X_{t+j} redundant but the learner may not be able to measure this decrease in uncertainty unless the knowledge of X_{t+i} is known. Thus, there is a tradeoff between effectiveness and time in choosing the value of I and it must be picked carefully by the engineer of the system.

In our experiments we also consider the variant of the algorithm shown in Figure 2.1 that we call *random sampling* in which instances are picked uniformly at random in all iterations. Random sampling is equivalent to traditional supervised learning. In the pseudo-code in Figure 2.1, random sampling corresponds to the case when $init\_size > T$.

2.2 Our Proposal: Feature Feedback and Instance Feedback in Tandem

We propose to extend the traditional active learning framework to engage the teacher in providing feedback on features in addition to instances. We describe the algorithm below and with pseudo-code in Figure 2.2. Steps 1 to 3.c are identical to the active learning system previously described. Our modifications to traditional active learning are in steps 3.d and 3.e. The active learner presents a list $\mathcal{P} = \{P_1 ... P_f\}$ of f features for the teacher

Tandem Learning

```
Input: $T$ (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), init_size
(number of random feedback iterations), $I$ (Number of instances labeled in each round)
Output: $\mathcal{M}_T$ (Model)

$t = 1; \mathcal{U}_0 = \mathcal{U}; \mathcal{M}_0$ =NULL;
1. While $t \leq$ init_size
   a.$\langle X_t, \mathcal{U}_t \rangle$ = Instance_Selection($\mathcal{M}_0, \mathcal{U}_{t-1}, random$)
   b. Teacher assigns label $Y_t$ to $X_t$
   c. $t++$
2. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t-1\}$, $\mathcal{M}_0$)
3. While $t \leq T$
   a. $\langle X_t, ...X_{t+I-1}, \mathcal{U}_{t+I-1} \rangle$=Instance_Selection($\mathcal{M}_t, \mathcal{U}_{t-1}, uncertain$)
   b. Teacher assigns label $Y_t...Y_{t+I-1}$ to $X_t...X_{t+I-1}$
   c. $\mathcal{M}_{t+I}$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t+I-1\}$, $\mathcal{M}_t$)
   d. i. $\{P_1, ..., P_f\}$ = Feature_Selection($\mathcal{M}_{t+I}, \mathcal{U}_t$)
      ii. Teacher selects $\mathcal{F} = \{F_1, .., F_k\} \subseteq \{P_1, ..., P_f\}$
   e. $\mathcal{M}_{t+I}$=Incorporate_Feature_Feedback($\mathcal{M}_{t+I}$, $\{F_1, ..., F_k\}$)
Return $\mathcal{M}_T$.
```

Figure 2.2. An active learning system where feedback on features is also requested (**System 2**).

to judge (step 3.d) at each iteration. P_i denotes the index of a feature ($1 \leq P_i \leq N$). Let $\mathcal{F} = \{F_1, ..., F_k\} \subseteq \mathcal{P}$ denote the subset of relevant features chosen by the user (in our final implementation in Chapter 6 the user also associates a label (± 1) with each F_i, but we ignore that for simplicity). The simplest implementation of such a system can consist of one where $\mathcal{F}$ is the union of all terms that occur in the I documents of the batch. This idea can be implemented as an interface where the user is asked to highlight relevant words, phrases or passages while reading the document in order to label the document (Step 3.b), akin to the system presented by Croft and Das [29]. Many times one would prefer to have the set $\mathcal{P}$ to be an ordered one, where feedback on P_1 is more valuable for the learner than feedback on P_2 and so on. We use such an approach in our final implementation in

Section 6.4. The user labeled features are incorporated in step 3.e where the model $\mathcal{M}_{t+I}$ is retrained with this added information.

In our proposed system the teacher is asked two types of questions: (1) membership queries and (2) questions about the relevance of features. A relevant feature is highly likely to help discriminate the positive class from the negative class. In this thesis we aim to determine whether a human teacher can answer the latter type of question sufficiently effectively so that active learning is accelerated significantly. A human and a classifier probably use very different processes to categorize instances. A human may use her understanding of the sentences within the document, which probably involves some reasoning and use of knowledge, in order to make the categorization decision, while a (statistical) classifier, certainly of the kind that we use in this thesis, simply uses patterns of occurrences of the features (phrases). Because of this difference it is not clear whether a human teacher *can* considerably accelerate the training of a classification algorithm by providing feedback on features.

Before we address that issue, we will show that feature feedback can accelerate active learning in an idealized setting (Chapter 5). We first seek to measure potential gain for improvement and then later examine how actual human teachers can approximate this ideal. Towards this goal we define an *oracle*. We use the oracle to obtain an upper bound on the performance of our proposed two-tiered approach. The oracle knows the correct answer needed by the learning algorithm. For example the word *ct* is a highly relevant feature for classifying Reuters news articles on the *earnings* category and our oracle would be able to determine that this feature is relevant when asked. However, a teacher (human) who did not understand that *ct* stood for *cents* may not be able to identify *ct* as relevant. Therefore, the oracle and teacher may differ in their answers to questions about features, that is, questions of type (2) above. We assume that the oracle and the teacher always agree on the labels of documents that is, questions of type (1) above. After showing the usefulness of oracle feature selection, we will develop algorithms for the Feature_Selection and Incorporate_Feature_Feedback routines for an SVM classifier (Chapter 6). We will then show that humans can emulate the oracle for feature feedback to an extent that results in significant improvements over traditional active learning (Chapter 7).

Note that the teacher is sometimes referred to as an *oracle* in the literature [11]. We will also use the term oracle to refer to a source that gives feedback on instances and/or features, but in this thesis we make a distinction between the teacher and the oracle. We will reserve the term teacher or user to refer to a real human, whose feedback may not be perfect, and we use the term oracle to refer to a source whose feedback is (close to) perfect for speeding active learning.

2.3 Summary

We developed the basic framework of the tandem learning algorithm in this chapter. The pseudo-code in Figure 2.2 will constantly be referred to throughout this thesis. We now move on to discuss the novelty and relevance of tandem learning in the context of past work in machine learning and information retrieval.

CHAPTER 3

RELATED WORK

Our work sits in the joint space between machine learning and interactive information retrieval. Sebastiani's survey paper [98] provides an overview of techniques in these two areas for solving text categorization. In this chapter we aim to place our work in context with previous research in these areas with respect to the following criterion:

- the kinds of feedback that have been considered in these areas and the success of those methods.
- whether a scenario of limited feedback has been considered.
- whether term and document feedback have been considered in conjunction with each other or in lieu of each other.

We describe approaches in machine learning and information retrieval, their advantages and disadvantages, stating how our work either compares with, or overcomes the deficiencies of past methods. We reserve describing other related work associated with choices of materials, methods or techniques (either our own or a choice made over existing ones) for when those choices have to be made. There is also a significant amount of literature in statistics which tries to determine the importance of feature selection when the number training examples is few [54]. We will relate our work to that work in Section 5.3.1.

3.1 Information Retrieval

We begin by discussing techniques for feedback in interactive information retrieval and then move on to discuss work that has been done with regard to interface design for interactive IR.

3.1.1 User Feedback in Information Retrieval

A typical task studied by the information retrieval community is ad-hoc retrieval [113] in which the user specifies her information need to a search engine through a query composed of a few keywords. The information retrieval community has studied various forms of feedback towards enhancing the representation of the original query with document feedback and term feedback being the chief forms studied [94].

A commonly used mode of feedback is pseudo-relevance feedback. The standard assumption made in this approach is that the top few retrieved documents are relevant [30, 69]. Then terms from these top documents are used to enhance the query. Although pseudo relevance feedback often works quite well, often improving the average performance of the system, it can sometimes hurt retrieval performance [55] due to the introduction of “noisy” terms into the initial query, typically due to the presence of few relevant documents in the initial retrieval. The latter may be due to a poor initial retrieval or due to the fact that there are very few relevant documents for the query in the corpus.

A more established alternative to pseudo-relevance feedback is “true relevance feedback” [90] in which a user is engaged in interactively providing the system with feedback on questions that the system generates. The user is typically asked to judge the relevance of documents or terms during feedback. True document feedback has been studied extensively [50] and is used in real world applications [1]. In the recent past it has been found that by using large external corpora on which to conduct the initial retrieval to enhance the query representation for pseudo-relevance feedback, the system can perform as well as by using true document feedback [38]. Basically, if the initial retrieval is guaranteed to be good with sufficiently many relevant highly ranked documents there is no need to ask for feedback and pseudo relevance feedback should suffice. Some past work has considered measuring the quality of the ranked list to determine whether feedback is necessary [4] using a clarity score [31]. Clarity may be considered to be a measure of "uncertainty" of a ranked list. Very few information retrieval techniques have considered asking for feedback on "uncertain documents" for ad-hoc retrieval. We will revisit this topic when we cover active learning in Section 3.2.1.

Term level feedback has been studied in information retrieval with mixed results [6, 29, 16]. The TREC interactive task focused on issues regarding the kinds of questions that can be asked of the user. Belkin et al. [16] found that users are willing to mark both positive and negative terms. Koenemann and Belkin [66] found that users are happy to use interfaces in which queries can be reformulated using a list of suggested terms and also found performance improvements for some tasks. However, Beaulieu [49] found that automatic query expansion using pseudo-relevance feedback resulted in better performance than term feedback by users. Kelly and Fu [62] had a similar experience with the TREC HARD track [114] but Diaz and Allan [37] found that term feedback helped improve performance on the same track.

In all of the above work the experimental setup has been such that it has been hard to ascertain whether the poor performance of an approach is due to algorithmic error or human error. In Chapter 1 we argued that in studying any kind of feedback one should question whether the feedback is necessary and useful by conducting upper bound experiments to measure the ability of the system's mechanism to absorb the feedback. Then one should question whether a human can provide the necessary feedback. Most of the interactive information retrieval experiments mentioned previously have an implicit hypothesis that term feedback will be useful and typically measure if users can provide feedback with little understanding of whether the term recommendation techniques and the methods for incorporating feedback are effective. More recently there has been work that tries to approach the ad-hoc retrieval task in a manner similar to ours by Magennis and Rijsbergen [78] and Ruthven [93]. Both papers found that users make sub-optimal decisions about terms to chose. Magennis and Rijsbergen had some obvious deficiencies in their approach, a fact that they acknowledge. Ruthven improved on the experimental setup of Magennis and Rijsbergen and found that term feedback using an oracle could give performance improvements even over automatic query expansion. However, he found that users cannot mark the terms required by the optimal query with reasonable precision. Oddly though, he did not report the performance achieved by the user-marked terms. In fact, we find that even though users marked only a fraction of the terms marked by the oracle, the performance improvements were on par with the oracle (Chapter 7).

Although both term and document feedback have been considered for ad-hoc retrieval, most of the work has been in understanding whether each of these modes of feedback individually help improve performance over the initial retrieval. Some feedback interfaces may have incorporated term and document feedback simultaneously [114], but there has been little work in factoring out the cost and benefit of one mode of feedback over the other. We also try to understand the kinds of problems for which a few quick rounds of feedback (at the term and/or document level) are likely to be useful. In fact we find that in scenarios of limited feedback a tandem approach is best.

In the ad-hoc retrieval tasks described above, feedback has been limited to a handful of rounds of either document or term feedback. In the TREC HARD track, the limitation is imposed by time; that is, the user gives feedback for a maximum of three minutes per query. However, there has been no systematic attempt to understand the "rate of learning" of different methods due to feedback like in our work. It must be noted that there has been some work in understanding the rate of learning due to relevance feedback [95], the purpose of those experiments was to determine the number of iterations for which relevance feedback showed significant improvements, rather than to compare multiple feedback algorithms to determine which algorithm results in a greater improvement in performance than the other.

A task more similar to the document classification task that we are interested in and one that the information retrieval community has worked on is filtering [72] . In the TREC filtering task, an initial information need is specified through a query. Subsequently documents are assumed to arrive in real-time and the system is expected to classify them as belonging to the topic of the query or not. In the TDT filtering (tracking) task [3], instead of an initial key-word query, the user is assumed to provide a sample document on the topic of interest. Again, the system is expected to classify documents arriving in a stream, on the basis of relevance to the topic. Both the TREC and TDT tasks have a "supervised" version where the user is assumed to give feedback on every document that the system classifies as relevant. The standard assumption of unlimited document feedback from the user in this task is an unreasonable one. In Section 7.5 we consider a more realistic version of the TREC filtering task, and find that term feedback is beneficial. Additionally in this task, like in ad-hoc retrieval, documents for which the user is asked for feedback, are typically ones that the system is highly confident about. Occasionally, the possibility of querying the user on uncertain documents, with the aim of exploration, in order to improve performance has been considered [117, 70]. However, there has been very little work that systematically tries to understand the effectiveness of such an approach and research in this direction in information retrieval is quite preliminary.

Thus there are myriad information seeking tasks studied by researchers in information retrieval for filtering with different starting points and different modes of feedback. However, as we have seen, a combination of term and document feedback with the aim of understanding whether one mode of feedback is better than the other has never been studied. Additionally little work has considered studying the impact of feedback on the rate of learning. Querying the user on uncertain documents as opposed to top-ranking documents is also not well-studied in information retrieval.

3.1.2 Human Computer Interaction

Human computer interaction (HCI) is an area of study that deals with interfaces between humans and computers. HCI for information retrieval takes into consideration the cognitive load of different interfaces for information access – query specification and refinement, visualization of search results and so on [66, 32, 87, 105]. Considering the cognitive load of interfaces is important, especially for an interactive technique like the proposed tandem learning system. Although our evaluation of the cognitive load of tandem learning in this work is by no means complete we have tried to bear in mind the willingness and ability of users to provide feedback throughout this work. Our preliminary user studies on the ability of users to mark terms versus documents and the time taken for each is interesting with several anecdotal examples that can possibly contribute towards a large scale user study which we leave for future work.

3.2 Machine Learning and Text Classification

We now review the machine learning literature, beginning with active learning, a foundational technique for our work.

3.2.1 Active Learning

Our proposed method is an instance of query-based learning [5] and an extension of standard ("pool-based") active learning which focuses on selective sampling of instances from a pool of unlabeled data [26]. The goal of active learning is to reduce the number of training examples needed to learn a classifier and therefore this learning paradigm fits well with our objective to learn quickly. To the best of our knowledge, all prior work on query learning and active learning focused on variants of membership queries, that is, requesting the label of a possibly synthesized instance. Our work is unique in the field of active learning as we extend the query model to include feature as well as document level feedback.

Query-based learning can be very powerful in theory [5], where the instance that the human is queried on is a synthetically generated uncertain example. Such arbitrary queries may be difficult to answer in practice. Baum and Lang [11] applied query-based learning to the problem of recognizing handwritten digits. The queries generated were often a random blur, halfway between two numbers, say a “5” and a “7”. Humans found this hard to label and hence the popularity of pool-based methods [26] that query the user on real instances from the corpus. We try not to let such a chasm between theory and practice exist in our work, by constantly considering the effectiveness and ease of predictive feature identification by humans in our application area – text classification.

Information retrieval has been using measures of uncertainty of the initial retrieval to determine whether feedback is necessary [4, 31]. However, those methods do not have the theoretical guarantees that methods studied in machine learning have. Secondly, they do not measure the uncertainty of an example (document) as is done in active learning. Whether standard active learning techniques will work for ad-hoc retrieval or vice-versa is a different question, and is a research question that is gaining popularity [101]. Our work is in text categorization for which there are uncertainty sampling measures from machine learning that are known to work well. Hence, we use those techniques for our basic architecture.

3.2.2 Explanation Based Learning

In explanation based learning, the learner is provided with one training example, a domain theory, some operational criteria and a goal concept [82, 36]. The explanation stage tries to use the domain information to prune away all unimportant aspects of a concept in order to explain the goal concept. The learned concept is then generalized, while keeping in mind the goal. Feature feedback may be viewed as the teacher providing evidence or an explanation for the learner on the reasoning behind the labeling. However explanation based learning is designed for deductive tasks rather than the inductive tasks that we are interested in.

3.2.3 Feature Selection

An engineer using a typical out-of-the-box machine learning system typically specifies a large set of features to represent instances, leaving the task of discerning good features from bad ones to the algorithm or to an automatic feature selection technique. Feature selection is often employed to improve the space or time efficiency of a classifier [21]. The impact on performance is dependent on the classifier and the number of training examples available [46]. When there are sufficient labeled instances, most state of the art learning algorithms are able to distinguish the relevant features from the irrelevant ones. Hence there is little improvement in performance with an additional feature selection component. Sometimes, for a classifier learned on ample training data, it is better to leave the feature selection to the learned weights and excessive feature pruning may even hurt performance [21]. However, when there are few labeled instances, working with a small set of relevant features tends to be significantly more useful. This phenomenon has been referred to in statistics as the Hughes phenomenon [53]. Unfortunately, to do automatic feature selection well, we need sufficient training data, leading to a chicken-and-egg problem. Other than the work by Hughes (which we will describe in detail in Section 5.3.1) we have seen little work in trying to understand the role of feature selection in scenarios with limited feedback, and particularly so for text. Our work appears to be the first to consider feature selection in an active learning setting. In Chapter 7 we show that users are able to do feature selection to an extent that results in sufficient enough overlap with an automatic feature selection algorithm trained on ample data, to result in a large improvement in performance by mitigating Hughes phenomenon.

3.2.4 Budgeted Learning

Budgeted learning is an area of machine learning that works on identifying the value of obtaining a measurement on a feature in determining the label of an instance. In a typical scenario the feature values are unknown and there is a cost to finding each feature's value for each instance of interest and the objective of the learner is to pick the most valuable feature to obtain a better estimate of the classifier (e.g., the work of Lizotte et al. [76]). For example, in trying to diagnose a disease, the doctor may have a choice between asking the patient to conduct one of many possible tests, each of which cost some money. Budgeted learning would help pick the most effective test to perform, by considering the cost and value of each feature towards decreasing uncertainty about the current diagnosis. In our setting, the cost for all features is the same and therefore, budgeted learning is not directly applicable.

3.2.5 Human Prior knowledge for Text Classification

Past work in text classification that has used human knowledge on features typically assumes that prior knowledge is given at the outset [115, 96, 35, 12]. The labels on features are typically used to "soft label" instances containing those features by assigning them to categories associated with those of the labeled features. This extra labeling is incorporated into training via modified boosting or SVM training. The classifier may use the "soft labels" by assigning low confidences to such instances or lower their misclassification costs compared to those of instances labeled directly by a human. One of our feature incorporation methods uses such a technique (Section 6.3.3). Soft labeling as an idea is almost identical to pseudo relevance feedback. Unlike the other work in text classification, we do not expect the user to be able to specify features from prior knowledge. We expect that our proposed interactive mode has an advantage over requesting prior knowledge from the outset, as it may be easier for the user to identify or recall relevant features while labeling documents in the collection and being presented with candidate features.

None of the works mentioned this far, consider the use of prior knowledge in the active (sequential) learning setting. The work of Godbole et al [47] is similar to ours in that it considers asking for features in an active setting. Their work however emphasizes system issues and focuses on multi-class training rather than a careful analysis of effects of feature selection and human efficacy. Their proposed method is attractive in that it treats features as single term documents that can be labeled by humans. Like the other work cited above, they also study labeling features before documents. They do not observe much improvements using their particular method over standard active learning in the single domain (Reuters) they test on. We enhance their simple idea in Section 6.3.3 and find improvements in performance.

Jones [59] also used single feature-set labeling in the context of active learning: the user was queried on a feature rather than the whole instance. The labeled feature was taken as a proxy for the label of any instance containing that feature, so labeling a single feature potentially labeled many documents (similar to the *soft* labeling technique discussed next). This was found to be more economical than whole-instance labeling for some tasks. The instances in their work consisted of only two features (a noun-phrase and a context), so labeling one feature is equivalent to labeling half an instance. Our work differs in that our instances (documents) contain many features (words) and we combine both feature labeling and document labeling.

Our study of the human factors (such as the quality of feedback and costs) is also a major differentiating theme between our work and previous work in incorporating prior knowledge for training. Past work has not addressed this issue, or might have assumed experts in machine learning taking a role in training the system [96, 115, 47, 59, 35]. We only assume a basic knowledge about the topic of interest.

3.3 Summary

We saw that the fields of machine learning, text classification and information retrieval have significant overlap in the kinds of problems they are trying to solve, with some overlap in techniques. How terms for feedback are obtained in text classification is typically naive where the user marks relevant terms at the outset, prior to learning. Often the human must go through a large list of features or must have sufficient domain knowledge to specify a complete list off the top of her head. There has been little consideration of techniques that solicit feature feedback in an “active” way. The information retrieval literature on the other hand, is rich in methods for term feedback, and some of these should be leveraged for term feedback in text categorization. Information retrieval research on feedback techniques has always kept interface issues in mind, therefore more easily bridging the gap between theory and practice. Machine learning on the other hand has much more variety in understanding how to choose examples for feedback through methods like active learning, whereas information retrieval has typically relied on top-ranking documents for feedback. We think a new approach which takes lessons from these two fields will result in much better performance for text categorization.

CHAPTER 4

METHODS AND MATERIALS

Much of this thesis is experimental and builds on certain standard text classification techniques like support vector machines (SVMs). We describe these techniques and algorithms in this chapter, diving into detail for SVMs. This level of detail is necessary to understand better the methods we used to explore the importance of feature selection (Chapter 5) and to understand the tandem learning algorithm developed in Chapter 6. We describe our data sets and evaluation measures in Sections 4.2 and 4.3. We also perform a sanity check of the basic building blocks of our system – SVMs and SVM active learning – in Section 4.4 to make sure they compare well with previously published results.

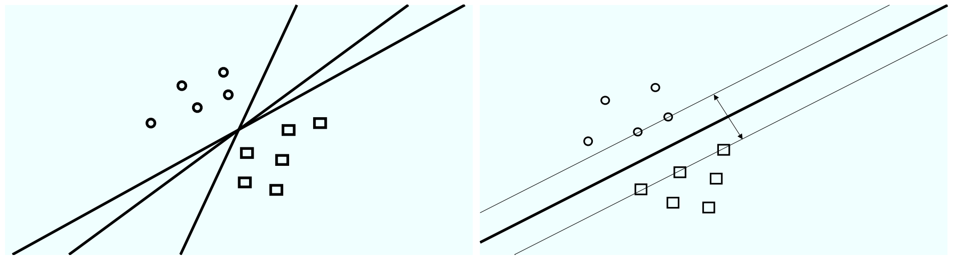

(a) Linear Classifier (b) Maximum margin Classifier

Figure 4.1. Linear classifiers.

4.1 Text Classification using Support Vector Machines

Support vector machines are a classification technique that have gained immense popularity in the recent past [112] and particularly so for text classification [57]. The main idea is to separate the classes by a hyper-plane that maximizes the margin between the two classes. Although the problem is formulated for linear hyper-planes, this is not a limitation as problems that are not intrinsically linear can be converted to linear problems through the use of a kernel [97]. Kernels transfer the problem to a high dimensional space where a linear hyper-plane is usually effective. A linear classifier is usually sufficient for text classification because of the intrinsic high dimensionality of text. We further motivate the choice of SVM as a base classifier in the next section, followed by the problem formulation for hard-margin and soft-margin classifiers. In the end of this section we describe active learning with support vector machines.

4.1.1 Why Support Vector Machines?

For many linearly separable problems there can be more than one hyperplane that separates the data (see Figure 4.1(a)). The simplest algorithm for a linear classifier is the *perceptron* algorithm [92]. The perceptron learns a linear function of the form $f(X) = w \cdot X + b$. 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 is trained fewer than $|\mathcal{T}|$ 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. The parameter η called the learning rate.

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 4.1(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 [112].

Support vector machines have been proven to be effective in many domains, and especially so for text classification and filtering [57, 22]. Although there has been some recent work that has shown that decision trees are better than SVMs on certain text classification tasks, and that SVMs match the performance of a decision tree only after aggressive feature selection [46], going by their generally good performance, we choose an SVM as the primary tool for investigation. Furthermore much research is being done for SVMs in the areas of incremental training of support vector machines [39] and concept drift [64], both of which are particularly useful given the dynamic content of document collections, allowing for future research that combines our techniques with these new ones. Uncertainty sampling with SVMs also provides a well understood framework for incorporating ideas from active learning [107].

4.1.2 Problem Formulation

Given a training set composed of t examples and the associated class information for each example, the objective is to obtain a hyperplane that best separates the data. More formally, the training set $\mathcal{T}$ consists of t example and class-label pairs ($\mathcal{T} = \{(X_1, Y_1)... (X_t, Y_t)\}$) (Refer to the train_classifier subroutines in Figures 2.1 and 2.2). Each X_i is represented as a vector, $\{x_{i,1}...x_{i,N}\}$, of N features. The classes belong to one of $\{+1, -1\}$ (i.e., $Y_i \in \pm 1$) with $+1$ denoting the label associated with an "on-topic" document and -1 denoting an "off-topic" document.

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

$$Y_i(w \cdot X_i + b) \geq 1 \; \forall i = 1...t \tag{4.1}$$

Note that if w and b are scaled by the same quantity, the decision surface given by Equation 4.1 is unchanged. A *canonical hyperplane* is defined as the hyperplane corresponding to the unique pair (w, b) when the following constraint is imposed:

$$\min_{i=1...t} |w \cdot X_i + b| = 1 \tag{4.2}$$

Normalizing the equation of the hyperplane in this way to its canonical form makes certain calculations convenient. The set of hyperplanes or (w, b) pairs that satisfy the constraints in equation 4.1 and 4.2 constitute the *version space*.

4.1.3 The Hard Margin Classifier

The width of the margin can be easily shown to be $2/|w|$ (in the canonicalized case) and given that the objective is to maximize this quantity, the optimization problem can now be stated as:

$$\begin{aligned} \min_{w,b} \phi(w) &= \frac{1}{2} w^T w \\ \text{subject to} \qquad & Y_i(w^T X_i + b) \geq 1 \quad i = 1...t \end{aligned} \tag{4.3}$$

The above optimization problem can be solved using quadratic programming by constructing the Lagrangian as follows:

$$L(w, b, \Lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{t} \lambda_i [Y_i(w \cdot X_i + b) - 1] \tag{4.4}$$

where $\Lambda = (\lambda_1, ..., \lambda_t)$ is a set of Lagrange multipliers corresponding to the constraints in Equation 4.3. More on the theory of Lagrange multipliers can be found in any standard convex optimization book [20]. Equation 4.4 has to be minimized with respect to w and b and maximized with respect to Λ ($\lambda_i \geq 0$). The solution can be found through standard quadratic programming optimization techniques which we will not outline here [97]. We use the implementation in the LibSVM toolkit [24].

To predict the class of an unlabeled instance Y_j, we simply compute the following quantity Y'_j (the prediction).

$$Y'_j = sgn(\sum_{i}^{t} Y_i \lambda_i X_i^T x_j + b) \tag{4.5}$$

This summation can be computed rather quickly because most of the λ_i values turn out to be zero. In fact the values are non-zero only for the *support vectors* – the instances that lie on the margin (See Figure 4.1(b)). The λ_i's control the extent of influence of each of the support vectors, with a more influential example having a greater value of λ_i.

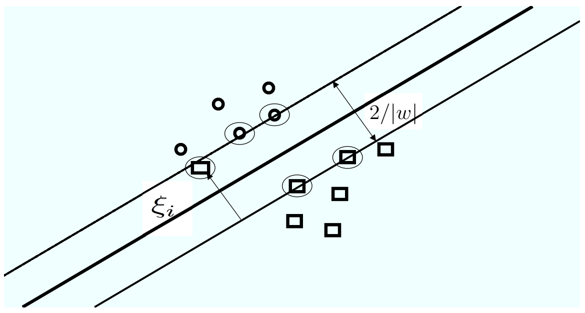


Figure 4.2. Soft Margin Classifier

4.1.4 Soft Margin Classifier

In reality all the constraints in Equation 4.3 will not be satisfiable, with data looking more like in Figure 4.2. In order to account for the violation of constraints, a set of t slack variables $\{\xi_i\}_{i=1}^{t}$, each corresponding to the classification error for a training instance is introduced. The optimization problem of 4.3 now becomes one of maximizing the margin (as before) and minimizing error. The modified problem is given as:

$$\begin{aligned} \min_{w,b,\Xi} \phi(w, \Xi) &= \frac{1}{2}||w||^2 + C\sum_{i=1}^{t}\xi_i \\ \text{subject to} \quad & Y_i(w \cdot X_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{4.6}$$

The constant C is a user specified constant, often referred to as the misclassification cost. C is typically obtained through cross-validation. A value of 10 is found to be effective for text-classification problems [57]. The Lagrangian for the soft margin classifier now becomes:

$$
\begin{aligned}
L(w, b, \Lambda, \Xi, \Gamma) &= \frac{1}{2}||w||^2 - \sum_{i=1}^{t} \lambda_i [Y_i(w \cdot X_i + b) - 1] \\
&+ \sum_{i=1}^{t} \gamma_i \xi_i + C(\sum_{i}^{t} \xi_i)
\end{aligned} \tag{4.7}
$$

The λ_i values are constrained to lie between 0 and C ($0 \leq \lambda_i \leq C$) for the soft margin classifier. As with the hard margin classifier, the λ_i values are non-zero only for support vectors. However, this time the values are bound (upper bound) by the user specified constant C. It turns out that we can easily specify a different C for different training examples, and in this way control the maximum influence of a given training example. The same QP solver in the LibSVM toolkit can be used for a problem where different examples have different misclassification costs. We will exploit this fact in the design of our algorithm in Chapter 6

4.1.5 Active Learning in Support Vector Machines: Uncertainty Sampling

Uncertainty sampling [75] is a 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 [107]. This is one way of implementing the *Instance_Selection* subroutine in Systems 1 and 2 (Figures 2.1 and 2.2 respectively). If an uncertain instance lies exactly on the hyperplane it results in a reduction of the version space by exactly half [107]. 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 much better than random sampling [75].

4.2 Data

Our test bed for this thesis comes from four standard domains. The first dataset consists of the 10 most frequent classes from the Reuters-21578 corpus [91]. The 12,902 documents are Reuters news articles categorized based on topics such as *earnings* and *acquisitions*. The Reuters corpus is a standard benchmark for text categorization.

The second corpus is the 20-Newsgroups dataset collected by Lang [68]. It has about 20,000 documents which are postings on 20 Usenet newsgroups. The corpus has a large vocabulary compared to the Reuters corpus (news articles tend to be more formal and terse) and it has many documents in each category which are tangentially related to the category's topic. The topics reside in a hierarchy with broader topics like *sports* and *computers* at the top level which are further divided into narrower subdivisions. For example, *sports* encompasses more focused groups like *baseball* and *hockey*. There are 20 categories at the lowest level of the hierarchy.

The third corpus is the TDT3 corpus [3] that has a 101K documents in 3 languages from both broadcast and news-wire sources. The Linguistic Data Consortium (LDC) provides the output of an automatic speech recognizer (ASR) for the broadcast news sources. Similarly for documents not originally in English they provide corresponding documents machine translated (MT) into English. We use the ASR and machine translated documents in our experiments in addition to the original English text. The noise in the ASR and machine translation output makes the TDT corpus particularly difficult to work with. The topics in the TDT corpus are based on news events. Thus, *hurricane Mitch* and *hurricane George* would be two different topics and developing a classifier to separate the two classes is seemingly a more difficult problem. The two classes would have a lot of common words especially with regard to lives lost, rescue operations etc. For example, the words *storm* and *damage* each respectively occur in 50% and 27% of the documents on *hurricane Mitch* and in 75% and 54% of the documents on *hurricane George*. These common words are probably useful to detect a generic topic like *hurricane* but are not that useful in discriminating *hurricane Mitch* from hurricane George. However, we think it would be fairly trivial for a human to point out *Mitch* and *George* as two keywords of importance which could then accelerate learning. The word *Mitch* occurs in 42% documents on *hurricane Mitch* and in 0 documents on *hurricane George*. Similarly, the word George appears in 0.05% documents on the topic of *hurricane Mitch* and in 88% of the documents on hurricane George.

The fourth corpus is the larger Reuters corpus [74] consisting of 810,000 documents in the English language covering a broad range of topics. The categories are labeled by Reuters and the labeling is not exhaustive, that is, only topics of interest to Reuters customers are labeled. There are a total of 104 categories, with each category being a node in a topic hierarchy. *Share listing*, *reserves* etc., are examples of topic categories.

One document can belong to multiple categories in the Reuters and 20 Newsgroups corpora, while in the TDT corpus a document can be associated with only one category. The Reuters-21578 corpus and 20 Newsgroups corpus are the main corpora we use for ablation experiments throughout. The final algorithm is also tested on the TDT3 and RCV1 corpora. For some of our experiments we used additional data sets. We describe them in the relevant chapters.

Documents were preprocessed using the Rainbow toolkit [81] to extract features by discarding stopwords and normalizing them (lowercase, stemming). Features that occurred fewer than 5 times in a dataset were discarded.

4.3 Evaluation

In this section we explore measures to quantify effectiveness and efficiency (the speed of learning) of active learning.

4.3.1 F1 for Effectiveness

The simplest and most popular metric for measuring text classification performance is *accuracy*, defined as

$$A = \frac{\text{number of documents correctly classified}}{\text{total number of documents}} \tag{4.8}$$

Many of the problems described in section 4.2 have high class skew, that is, the ratio of positive class to negative class for our problems ranges from 0.01% to 42%. For a class skew as low as 0.01%, a naive classifier that classifies all instances as negative achieves almost 100% accuracy. Therefore, we need a metric that concentrates on how much of the relevant material has been filtered. One such measure is *precision* (P) which takes into account how many of the documents declared on-topic were truly relevant and is given as:

$$P = \frac{\text{number of relevant documents detected}}{\text{number of documents detected on topic}} \tag{4.9}$$

The second measure *recall* (R) measures how many of the relevant documents were actually detected to be on topic.

$$R = \frac{\text{number of relevant documents detected}}{\text{number of relevant documents}} \tag{4.10}$$

There is an inverse relationship between P and R: a technique that predicts very few positives will have high precision, but low recall and vice-versa. We therefore cannot use precision or recall alone and need a measure that combines the two. $F1$ is one such measure and is given as:

$$F1 = 2PR/(P + R) \tag{4.11}$$

We denote effectiveness of a model trained on t training examples by $F1_t$. In our experiments t is typically small, taking values as low as 2, 7, 12 and so on.

4.3.2 Efficiency

The *deficiency* measure was proposed by Baram et al [10] as a measure of the speed of an active learning algorithm, with the aim of comparing different active learning algorithms. Baram et al. defined deficiency in terms of accuracy. Since accuracy is not a reasonable measure of performance for the classification problems we have chosen, we modify the definition of deficiency, and define it in terms of the $F1$ score. For deficiency a lower value is better. As we also report on the $F1$ scores, for which higher values are better, for consistency and easier interpretation of our charts and tables we define *efficiency* $= 1-$*deficiency*. Efficiency has a range from 0 to 1, and a larger value indicates a faster rate of learning. Thus, in all our reports higher values are better.

Let $F1_t(\text{RAND})$ be the average $F1$ achieved by an algorithm when it is trained on t randomly picked instances and $F1_t(\text{ACT})$ be the average $F1$ obtained using t actively picked instances.

Efficiency, E_T is defined as:

$$E_T = 1 - \frac{\sum_{t=2}^{T}(F1_M(\text{RAND}) - F1_t(\text{ACT}))}{\sum_{t=2}^{T}(F1_M(\text{RAND}) - F1_t(\text{RAND}))}$$

$F1_M(\text{RAND})$ is the $F1$ obtained with a large number (M) of randomly picked instances. The value $F1_M(\text{RAND})$ represents the performance of a classifier with a large amount of training data, and can be considered the optimal performance under supervised learning. With large amounts of training data, we expect the performance of a classifier trained using active learning to be about the same as a classifier trained using random sampling. However, we would like active learning to approach this level as *quickly* as possible. The metric therefore takes into consideration how far the performance is from the optimal performance by computing the difference $F1_M(\text{RAND}) - F1_t(\text{ACT})$ and $F1_M(\text{RAND}) - F1_t(\text{RAND})$. The metric compares this difference when t documents have been actively picked to the difference when t documents have been randomly picked for increasing number of training documents t.

Since we are concerned with the beginning of the learning curve, we stop after $T = 42$ number of documents have been sampled. For expedience, we did not measure performance at every point from 2 to 42 labeled documents, but compute the summation at discrete intervals, measuring $F1$ after each additional five documents have been labeled: $t = 2, 7, 12, 17...42$. For this thesis we take $M = 1000$, that is, we consider the optimal random-learning performance to be attained after the classifier has seen 1000 labeled instances. In our experiments $F1_t(\bullet)$ is the average $F1$ computed over 10 trials. In addition to efficiency we report $F1_t$ for some values of t.

To understand the intuition behind efficiency, we can draw the active learning curve by plotting $F1_t(\text{ACT})$ for increasing values of t, as shown in Figure 4.3. Similarly we can draw the random learning curve by measuring $F1_t(\text{RAND})$ for increasing values of t. $F1_M$ is a straight line representing the best achievable performance. Then efficiency is one minus the ratio of the solid colored area to the spotted area. The higher the efficiency, the better the active learning algorithm. We aim to maximize both efficiency and $F1$. In some of our experiments we obtain efficiencies exceeding 1. This is due to using a finite M: it is possible that a classifier produced by active learning on 42 or fewer instances may do better than a classifier trained on a random sample of a 1000 instances.

4.4 Testing the Base Classifier

We compared the base SVM classifier and the implementation of uncertainty sampling we use to previously published baselines on the 20 newsgroups corpus. On conducting experiments similar to those of Bekkerman et al. [13] we obtain a Macro averaged BEP (Precision at the point where Precision=Recall) of 82.8% for the 20 newsgroups corpus. Bekkerman et al., obtained a slightly higher BEP of 85.6% (See Table 5 of their paper). However, one must note that they employed feature selection which we did not. In spite of that, the performance is comparable, indicating that SVMs are robust in the presence of many irrelevant and redundant features.

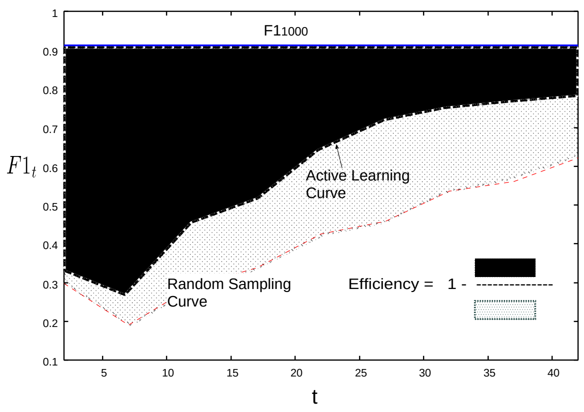


Figure 4.3. The figure illustrates *efficiency*, the performance metric which captures rate of learning. The figure on the right illustrates the *learning surface*. The plot is a measure of $F1$ as a function of the number of features and training documents. The dotted line traces the region of maximum $F1$. With few training documents, aggressive feature selection (few features) is needed to maintain high accuracy. The thick dark band illustrates traditional active learning.

We then compared the performance of vanilla uncertainty sampling (active learning) from Tong's dissertation [108]. We try to emulate the experimental conditions described in Section 4.1.3 of that work. Uncertainty sampling with SVMs is called the "Simple" method in that work. The results of our implementation (Figure 4.4) compare quite well with theirs (Figure 4.5 in Tong's thesis [108]).

4.5 Summary

In this Chapter we reviewed methods and materials used in this thesis. We use support vector machines, uncertainty sampling, and fairly standard preprocessing to build an underlying framework which compares well with the state of the art in text classification and active learning. We also outlined the measures used to quantify the performance of our methods laying out the complete experimental framework for this thesis.

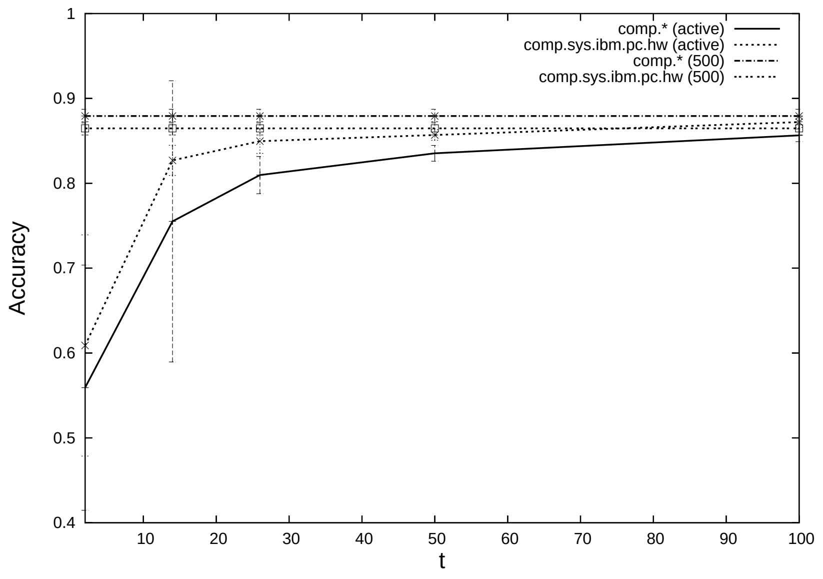


Figure 4.4. Comparing the base classifier with the state-of-the-art.

CHAPTER 5

ORACLE EXPERIMENTS

Equipped with the intuitions developed in Chapter 1, and the tools developed in Chapter 3, we now explore the benefits of human aided feature selection in active learning. We leave the question of how those features may be obtained for later (Section 6.2) and rather try to determine the extent of the gains we can get from feature selection using an oracle in Section 5.2. We then explore why we obtain these gains by asking how feature selection impacts the underlying learning algorithm in Section 5.3.

In this Chapter we seek the answer the following questions:

- *Can feature feedback significantly boost active learning performance?* In Section 5.2 we find that feature feedback using an oracle helps improve active learning performance, especially in the early stages of learning, when the number of labeled documents is as few as 7 and 22. The impact of feature selection decreases as the number of labeled examples increases.

- *Should we use feature feedback during the entire active learning process (both instance selection, and model selection) or only for model selection?* In Section 5.3.2 we find that most of the benefit of feature feedback is for model selection, although there is some benefit for instance selection.

5.1 Design of the Approximate Feature Oracle

The (feature) oracle in our experiments has access to the labels of all documents in the data-set (hence the name oracle) and uses this information to return a ranked list of features sorted in decreasing order of importance. Information gain is a common measure for ranking features and has been found to be quite effective [98, 21], and is easy and quick to compute. 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) from section 4.1, and τ is 0 or 1 indicating the presence or absence of a feature respectively. In the oracle experiments in this chapter, we cut off the ranked list (therefore obtaining a feature subset) at the point that yields the highest average active learning performance.

5.2 Extent of Speed Up Possible: Oracle Experiments

Following the algorithm for System 3 (Figure 5.1), let $f = N$ (the total number of features) and let us assume that the oracle selects the k most important features (by information gain) in Step 1.b. The set $\mathcal{F}$ is used to initialize the model in Step 2. In our implementation this initialization informs the model to ignore all features not in $\mathcal{F}$ during training. Random sampling (Step 3.a), in this particular implementation, does not use any of the feature information or the initial model. The model is then trained on the initially picked instances (Step 4). Since the model has been initialized with the set $\mathcal{F}$, the train_classifier subroutine zeros out the values of all features not in $\mathcal{F}$ for each of the training instances $X_i, 1 \leq i \leq t$. We now perform active learning on the instances in this reduced feature space (Step 5). We evaluate these experiments at many points in the two-dimensional space of number of features k versus number of labeled documents t by measuring the F1 score: $F1_t(\text{ACT}, k)$[1]. We can similarly measure performance in the reduced feature space when instances are

[1]Note the slight difference in notation from Chapter 4. The additional parameter k denotes the number of features used by the classifier. Note that $F1_t(\text{ACT}) = F1_t(\text{ACT}, N)$

picked randomly. Thus we can compute efficiency in the reduced feature space as $E_T(k)$. When $f = k = N$ the algorithm reduces to traditional active learning (Figure 2.1).

When System 3 is used with a user instead of the oracle it is equivalent to a scenario where prior knowledge is used to initialize the classifier [96, 115, 47, 59].

Use of Feature Feedback Before Active Learning

Input: T (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances), *init_size* (number of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

```
$t = 1; \mathcal{U}_0 = \mathcal{U}$; $\mathcal{M}_0$ =NULL;
1.a. $\{P_1, ..., P_f\}$ = Feature_Selection($\mathcal{U}_0$)
  b. Oracle selects $\mathcal{F} = \{F1, .., F_k\} \subseteq \{P_1, ..., P_f\}$
2.$\mathcal{M}_0$=Incorporate_Feature_Feedback($\mathcal{M}_0$, $\{F_1, ..., F_k\}$)
3. While $t \leq$ init_size
    a. $\langle X_t, \mathcal{U}_t \rangle$=Instance_Selection($\mathcal{M}_{t-1}, \mathcal{U}_{t-1}, random$)
    b. Oracle assigns label $Y_t$ to $X_t$
    c. $t++$
4. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_0$)
5. While $t \leq T$
    a. $\langle X_t, \mathcal{U}_t \rangle$=Instance_Selection($\mathcal{M}_{t-1}, \mathcal{U}_{t-1}, uncertain$)
    b. Oracle assigns label $Y_t$ to $X_t$
    c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i \rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
    d. $t++$
Return $\mathcal{M}_T$
```

Figure 5.1. An active learning system where feature selection is done before instance selection (**System 3**). This is one of the two set-ups used in our oracle experiments described in Section 5.2. The second set-up is shown in Figure 5.7.

5.2.1 Experimental Setup

We performed our experiments on the Reuters-21578 and 20 Newsgroups corpora. We consider each topic as a one-versus-rest classification problem, giving us a total of 30 such problems (listed in Appendix A). We also pick two pairs of easily confusable classes from the 20-Newsgroups domain to obtain two binary classification problems viz., *baseball* *vs hockey* and *automobiles vs motorcycles*. In all we have 32 classification problems for experiments in this chapter.

5.2.2 Improvements to Active Learning with Feature Selection

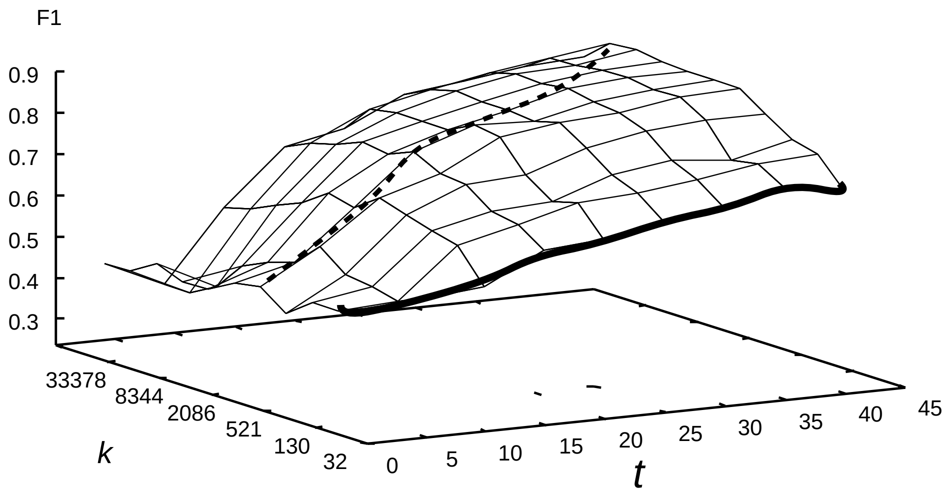


Figure 5.2. The figure illustrates the *learning surface*. The plot is a measure of $F1$ as a function of the number of features (k) and training documents (t). The dotted line traces the region of maximum $F1$. With few training documents, aggressive feature selection (few features) are needed to maintain high accuracy. The thick dark band illustrates traditional active learning.

Figure 5.2 shows a plot of $F1_t(\text{ACT}, k)$ for different values of the number of features k and number of labeled training instances t, for the *earnings* category in Reuters. The dotted curve traces the maximum F_t for each value of t. The x, y and z axes denote k, t and $F1_t(\text{ACT}, k)$ respectively. The number of labeled training instances t ranges from 2 to 42 in increments of 5. The number of features used for classification k has values from $33,378$ (all features), $33378/2$, $33378/4$ to 32. The dark band represents the case when all features are used. This method of learning in one dimension is representative of traditional active learning. Clearly when the number of documents is few, performance is better when there is a smaller number of features. As the number of documents increases the number of features needed to maintain high accuracy increases. From the figure it is obvious that we can get a big boost in accuracy by starting with fewer features and then increasing the complexity of the model as the number of labeled documents increase.

Table 5.1 captures the behavior of all the problems in the Reuters corpus when there is an oracle to do the feature selection. The second column ($k = N$) in Table 5.1 shows the efficiency obtained using uncertainty sampling and all N features. The third column ($k = n$) indicates the average efficiency obtained using uncertainty sampling and a reduced subset of n features. The feature set size n at which this efficiency is attained is shown in column four. For each classification problem, we identify the feature set size which optimizes the efficiency, that is, optimizes the rate at which classification performance under active learning approaches learning with all of the data. This optimal feature set size for active learning n is given by

$$n = \text{argmax}_k E_{42}(k)$$

Figure 5.3 shows the efficiencies at $E_{42}(N)$ and $E_{42}(n)$ for the individual problems in the three corpora. In many cases, $E_{42}(N)$ is much less than $E_{42}(n)$.

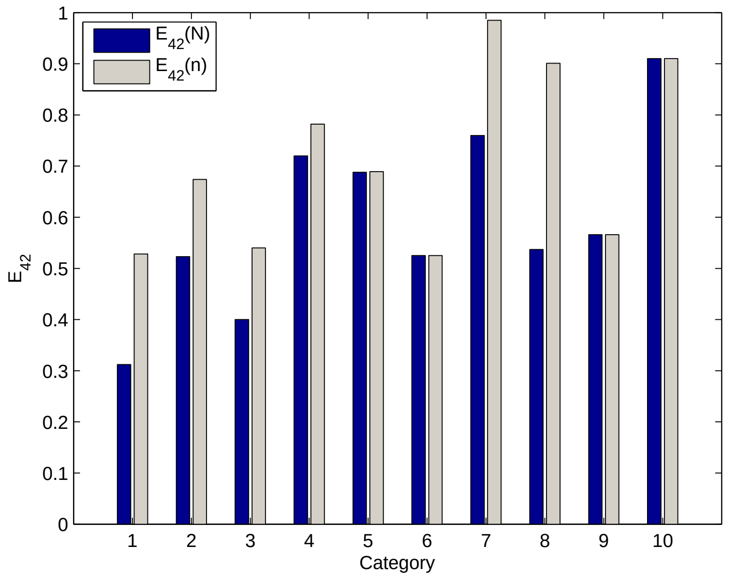
Column 5 ($k = N$) in Table 5.1 shows the value of $F1_7(\text{ACT}, N)$: the F1 score with seven instances selected using active learning, when all features are used. Column 6 shows the average $F1_7(\text{ACT}, m)$ using a reduced feature subset. As for efficiency, the best feature subset size (m) for each classification problem is obtained as the feature subset size at which $F1_7(\text{ACT}, k)$ is maximum. For example in Figure 5.2 at 7 instances the best $F1$ is obtained with 512 features. Figure 5.4 shows the values of $F1_7$ computed using all (N) features and using a reduced subset of (m) features for individual problems.

Columns 7, 8, and 9 in Table 5.1 show similar results for $F1_{22}(\text{ACT}, k)$ with the best feature subset size at $t = 22$ being denoted by p. The values for individual problems is illustrated in Figure 5.5. The last column shows $F1_{1000}(\text{RAND})$.

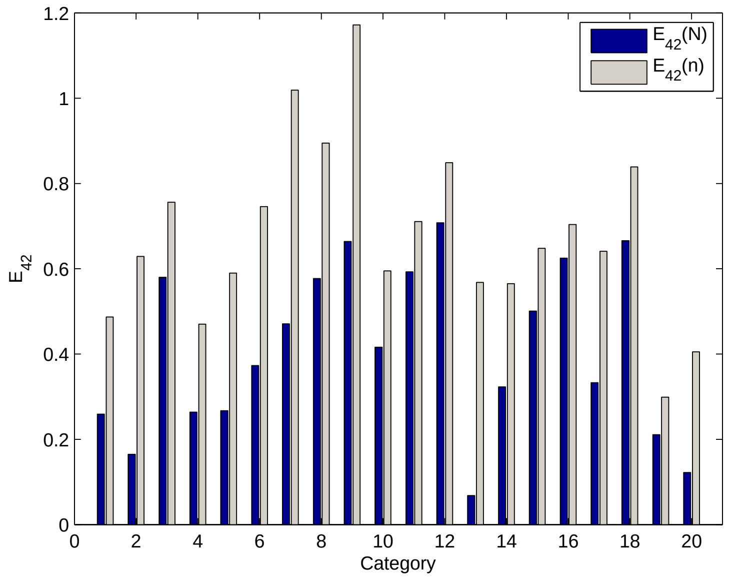
All 32 of our classification problems exhibit behavior as in Figure 5.2. For all classification problems, n, m and p are less than the maximum number of features. In most

Dataset	$E_{42}(k)$			$F1_7(\text{ACT}, k)$			$F1_{22}(\text{ACT}, k)$			$F1_{1000}$
↓	$k = N$	$k = n$	n	$k = N$	$k = m$	m	$k = N$	$k = p$	p	
Reuters	0.59	**0.68**	11179.3	0.36	**0.48**	8481.1	0.580	**0.66**	11851.6	0.73
20 NG	0.40	**0.66**	41.5	0.07	**0.22**	48.3	0.21	**0.29**	487.1	0.45
Bas vs Hock	0.29	**0.55**	25	0.59	**0.70**	25	0.78	**0.83**	200	0.96
Auto vs Mot.	0.68	**0.32**	125	0.43	**0.72**	62	0.76	**0.86**	31	0.90

Table 5.1. Improvements in efficiency, $F1_7$ and $F1_{22}$ using an oracle to select the most important features (Figure 5.1). We show results for each metric at N (total number of features for a particular dataset) and at feature set sizes for which the scores are maximized (n, m and p for E_{42}, F_7, and F_{22} respectively). For each of the three metrics, figures in bold are statistically significant improvements over uncertainty sampling using all features (the corresponding columns with feature set size of N). We see that with only 7 documents labeled ($F1_7$) the optimal number of features is smaller (8481.1 on average), while with more documents labeled, (22 for $F1_{22}$) the optimal number of features is larger (11851.6 on average). When 1000 documents are labeled ($F1_{1000}$) using the entire feature set leads to better scores with the $F1$ measure. This suggests that our best active-learning algorithm would adjust the feature set size according to the number of training documents available. Numbers in bold are statistically significant (using a two-tailed t-test at the 0.05 level of alpha) over the preceding column.

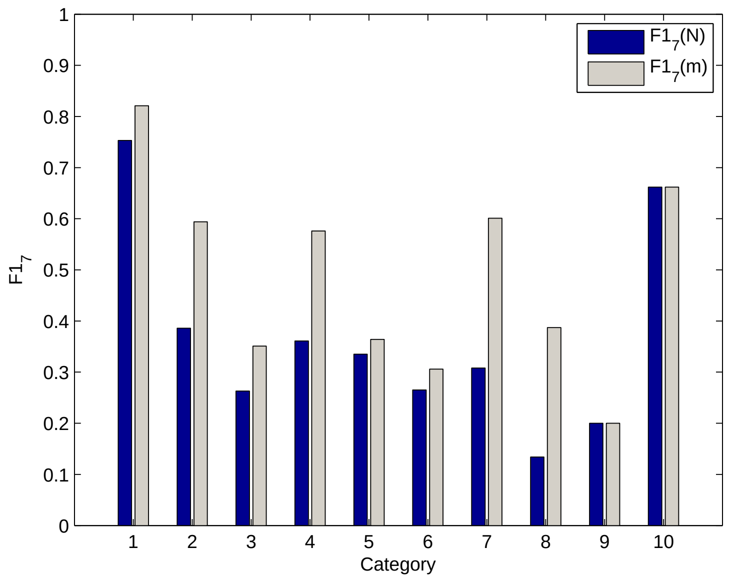


(a) Reuters

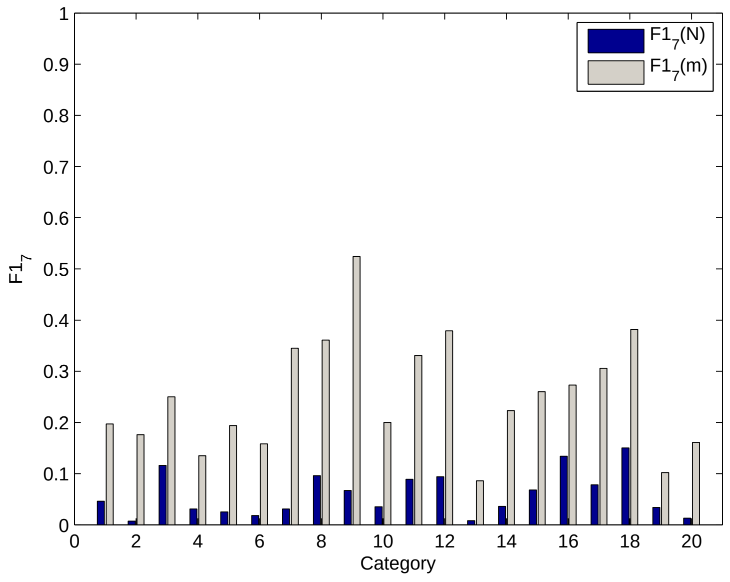


(b) 20 Newsgroups

Figure 5.3. Improvements in efficiency using an oracle to select the most important features. For each problem we show efficiency at N (total number of features for a particular dataset) on the right and efficiency at the feature set sizes for which the efficiency is maximized (n) on the left. The class keys are given in Appendix A.

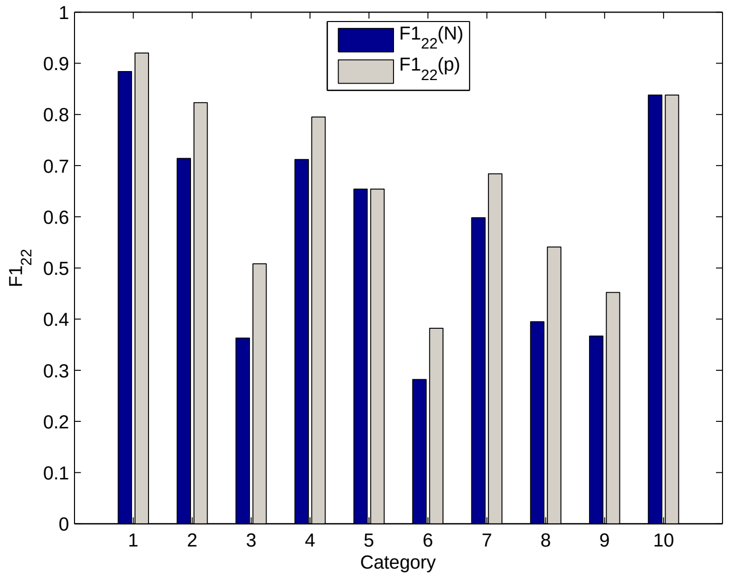


(a) Reuters

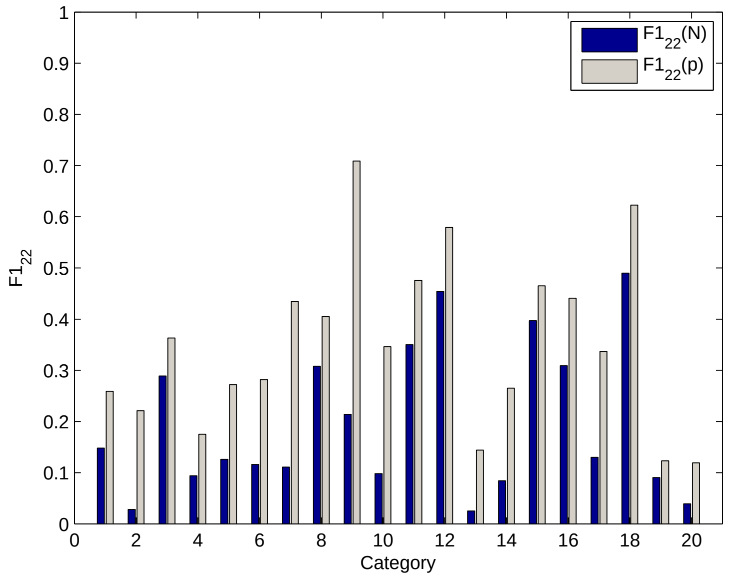


(b) 20 Newsgroups

Figure 5.4. Improvements in $F1_7$ using an oracle to select the most important features. For each problem we show $F1_7$ at N (total number of features for a particular dataset) on the left and $F1_7$ at the feature set sizes for which the $F1_7$ is maximized (m) on the right. Remember, the objective is to maximize $F1_7$. The class keys are given in Appendix A.



(a) Reuters



(b) 20 Newsgroups

Figure 5.5. Improvements in $F1_{22}$ using an oracle to select the most important features. For each problem we show $F1_{22}$ at N (total number of features for a particular dataset) on the right and $F1_{22}$ at the feature set sizes for which the $F1_{22}$ is maximized (p). Remember that the objective is to maximize $F1_{22}$. The class keys are given in Appendix A.

cases $m \leq p$ (that is, the number of features optimal for 7 labeled instances, m is less than the number of features optimal for 22 labeled instances, p) meaning that as the number of labeled instances (t) increases, the complexity of the classifier also needs to increase. For 20-Newsgroups, for all classes we observe that efficiency, $F1_7$ and $F1_{22}$ are best at very small feature subset sizes. For Reuters there are classes for which a large number of features become important very early (for example: *trade*).

5.3 Why is feature selection useful?

In the next sub-section we try to understand our results of the previous section – that with few training examples, few well selected features gives the optimal accuracy, and that with increase in the amount of training data feature selection becomes less important – in the context of similar research in statistics. We will then conduct experiments to determine the part of the active learning process that feature selection is aiding and thereby boosting classifier performance.

5.3.1 Mitigating Hughes Phenomenon

There exists plenty of literature in statistics that shows that when there are few training samples (t is small), as more features are added (k is increased), then the accuracy of the classifier increases up to a point after which it starts decreasing [54]. If t is large, as more features are added, the accuracy increases up to a point and then shows no change. This is called Hughes' phenomenon [53] and it suggests that when the number of training instances is small, the system needs to use a smaller number of features. This problem has also been referred to as the "curse of dimensionality" [17].

Jain and Chandrasekaran [54] provide a comprehensive overview of the early research in statistics that takes into account dimensionality and sample size considerations in pattern recognition. They first discuss the peaking of classifier performance shown by Allais [2] for linear classifiers. Consider that $f(X)$ is a linear predictor of Y of the form $f(X) =$

$w \cdot X + b$. Allais assumed that the joint distribution of Y and X was multivariate Gaussian and a maximmum likelihood (ML) estimator $g(X)$. He compared the mean squared error (MSE) of the ML predictor with that of an ideal predictor. The MSE is given as: $\epsilon = E[Y - g(X)]^2$. Allais then derived the expectation of the MSE as function of the number of training examples (t) and the number of features ($k, k \leq N$) used to represent the data.

$$\begin{aligned} E(\epsilon) &= \delta^2(\frac{t+1}{t})(1 + \frac{k}{t-k-2}) \quad \text{for } k \leq t-2 \\ &= \infty \qquad\qquad\qquad\qquad\quad \text{for } k = t-1 \\ &= undefined \qquad\qquad\qquad \text{for } k \geq t \end{aligned} \tag{5.1}$$

δ is the ideal MSE. The fractional increase in expected MSE when the sample is of size t is approximately $k/(t-k-2)$. When $k << t$ the fractional increase is k/t. It is clear that one would want that $k << t$, which is similar to our empirically arrived conclusion of the previous section. The question then is whether equation 5.1 is relevant when the assumptions about the estimation procedure and the data distribution made by Allias are removed.

Hughes [53] then studied the behavior of a finite-sample Bayesian classifier with increasing dimensionality. In the Bayesian setting, the addition of features can only add to the information of a classifier, and as long as the old set of features is completely recoverable from the new set of features, there should be no decrease in accuracy. Hughes assumed a two-class problem and a discrete pattern measurement environment. Other than that, no Gaussian or statistical independence assumptions were made. Hughes estimated a quantity called the mean accuracy, a function of t and k. Assuming the apriori distribution of each class to be equal, he found that, for a fixed value of t, as k increased, the mean accuracy increased till some value $k = k_{opt}$, after which it started decreasing. When $t = \infty$, we have complete information, and as expected, peaking does not occur. Figure 5.6 illustrates

Hughes' phenomenon by plotting his estimated mean accuracy for different values of t, for the simple case when the apriori distributions of both classes are equal.

Hughes' result led to a number of investigations and while there have been arguments about the estimated mean accuracy according the Hughes' formula (plotted in figure 5.6), it is generally accepted that peaking occurs in classifiers [60]. Foley [44] then went on to investigate whether there is any general rule of thumb k/t ratio that the designer of a pattern recognition system could use. He demonstrated that the training set error rate is an extremely biased estimate of the test set error rate, if the ratio k/t is less than three. His results which are empirical, are based on the assumption that the underlying distribution is Gaussian.

Hughes' phenomenon can also be explained using the bias variance tradeoff. With a finite number of training examples, when a new feature is added, the Bayes error decreases, but now more parameters need to be estimated from the same number of samples. This increases variance. The degradation in performance due to the addition of features is because the increase in variance is more than the decrease in the Bayes classification error.

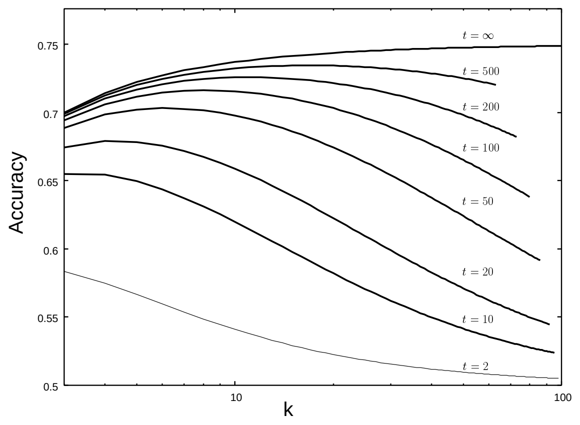


Figure 5.6. Mean accuracy falls with increasing k.

The problem with all past work studying Hughes phenomenon in statistics is that they have either used synthetic data [44] and have made certain assumptions about the underlying distributions. Whether such behavior is demonstrated in natural domains like text and images is unanswered. The work of Shahshahani et al [99] is similar to ours, in that they investigate Hughes phenomenon in a real world domain, albeit a different one. Their work is in the domain of remote sensing, where the data consisting of spectral images of the earth needs to be classified into categories such as soil, wheat etc. They use the spectral bands of the images as features and address the question whether fewer bands (smaller k) need to be used when t is small. As expected, they find the answer to be in the affirmative. In addition, they showed that Hughes' phenomenon can be mitigated in the presence of unlabeled data.

The advantage of Shahshahani et al.'s domain was that they already had an ordering determining the importance of features (the first spectral band has the most information, the second less, and so on), hence they did not have to resort to explicit feature selection. In text we have no such natural ordering. We used the oracle to generate the required ordering and demonstrated that Hughes' phenomenon does indeed occur in domains of text classification. Another difference from all previous work (both the statistics literature and that of Shahshahani et al [99]) is that we demonstrated Hughes phenomenon in an active learning setting. In our setting there is access to some (not all) of the unlabeled data (the pool $\mathcal{U}$). Hughes phenomenon still occurs. Therefore in filtering-like applications where new data keeps arriving and we do not have access to such a large volume of unlabeled data, we still need an oracle for feature selection.

5.3.2 Feature Selection: Aiding Model Selection or Instance Selection?

In the system in Figure 5.1 feature selection is done prior to active learning. We now implement a simple variation of that experiment. In this experiment, active learning proceeds normally with all the features available, but after all the instances are picked (after T iterations), the best set of k features, that improve the resulting trained classifier the most, are picked and the resulting performance is reported. This is shown schematically and with pseudo-code in Figure 5.7. We note that even when starting with the same initial set of labeled instances, the classifiers learned during active learning, hyperplanes in our

Use of Feature Feedback After Active Learning

```
Input: $T$ (Total number of feedback iterations), $\mathcal{U}$ (Pool of unlabeled instances, init_size (number
of random feedback iterations)
Output: $\mathcal{M}_T$ (Model)

$t = 1; \mathcal{U}_0 = \mathcal{U}; \mathcal{M}_0 =$NULL;
1. While $t \leq$ init_size
   a. $X_t$ = Instance_Selection($\mathcal{M}_0, \mathcal{U}_{t-1}, random)$
   b. Oracle assigns label $Y_t$ to $X_t$
   c. $t++$
2. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i\rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
3. While $t \leq T$
   a. $\langle X_t, \mathcal{U}_t\rangle$ = Instance_Selection($\mathcal{M}_{t-1}, \mathcal{U}_{t-1}, instance)$
   b. Oracle assigns label $Y_t$ to $X_t$
   c. $\mathcal{M}_t$ = train_classifier($\{\langle X_i, Y_i\rangle | i = 1...t\}$, $\mathcal{M}_{t-1}$)
   d. $t++$
4. a. $\{P_1, ..., P_f\}$ = Feature_Selection($\mathcal{M}_T, \mathcal{U}_T$)
   b. Oracle selects $\mathcal{F} = \{F1, .., F_k\} \subseteq \{P_1, ..., P_f\}$
5. $\mathcal{M}_T$=Incorporate_Feature_Feedback($\mathcal{M}_T$, $\{F_1, ..., F_k\}$)

Return $\mathcal{M}_T$
```

Figure 5.7. An active learning system where feature selection is done after instance selection (**System 4**). This is one of the two set-ups used in our oracle experiments described in Section 5.2. The first set-up is shown in Figure 5.1.

case, in the Systems 3 and 4 may be different as they are learned in different spaces (using different feature subset sizes). Besides, the set of labeled instances is small, so the learning algorithm may not be able to find the best "unique" hyperplane. In turn, the instances picked subsequently during active learning may differ substantially as both the spaces the instances reside in and the learned classifiers may be different. The classifier learned in the feature reduced space may have better accuracy or lead to better choice of instances to label during active learning, though this is not guaranteed or the benefits may be negligible. In short, the trajectory of the active learning process, that is, the instances labeled and classifiers learned, can be different in the two regimes, which may lead to substantially different active learning performance. In the next section we provide the details of these experiments.

The difference between Systems 3 and 4 is in that feature selection precedes active learning in the former, and the best feature subset is picked in a retrospective manner, while it follows active learning in the latter. The two systems when used with oracle feature selection will help us understand the extent to which oracle feedback aids different aspects of the active learning process. Figure 5.8 compares the results of using System 4 and system 3 on the Reuters corpus.

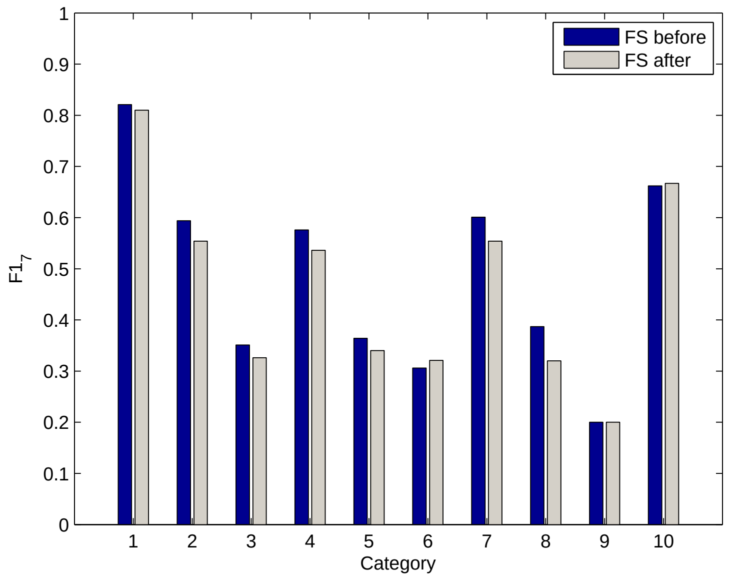
There is hardly any difference between systems 3 and 4, especially on $F1_7$. All other datasets exhibit the same behavior. The $F1_{22}$ and E_{42} values are slightly better for the method that does feature selection before active learning (system 3) but it is not significantly different (determined using a t-test at the 95% level of confidence) from the method where feature pruning is done after instance selection (system 4). Thus, our experimental results suggest that there is some benefit for instance selection but most of the benefit from oracle feature selection comes from improving the model learned (model selection).

Hypothesis 1

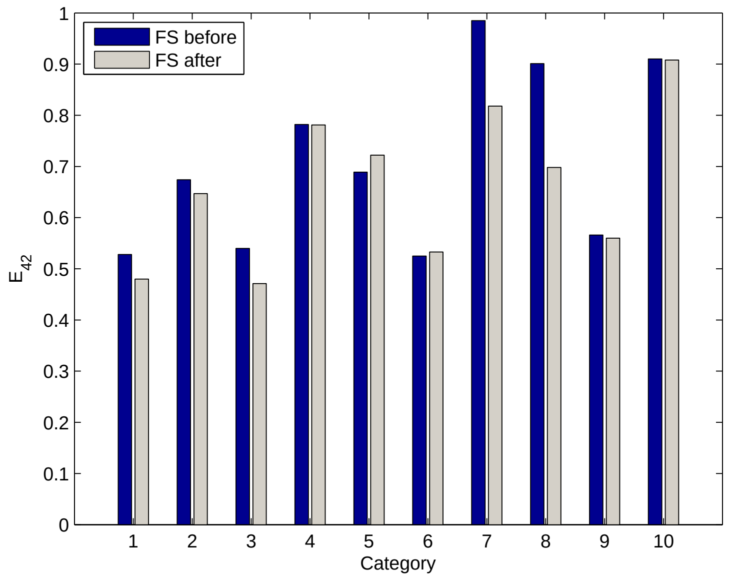
We showed that aggressive feature selection was needed in the early stages of learning. We also showed that most of the benefit due to feature selection was in model selection though there was some benefit for instance selection in the active learning process as well. At this point we have shown sufficient evidence to prove the first of our three hypotheses (Section 1.5) – that there exists a set of features for which if the learner is provided relevance information, the speed of active learning can be significantly improved.

5.4 Summary

With limited labeled data, there is little evidence to prefer one feature over another, so the learner has to spread the feature weights more or less evenly on many features. In other



(a) $F1_7$



(b) E_{42}

Figure 5.8. $F1_7$, $F1_{22}$ and efficiency E_{42} for the Reuters corpus when feature selection is done before active learning (system 3) and when feature selection is done after active learning (System 4).

words, the learner has to remain conservative. Feature/dimension reduction by the oracle allows the learner to “focus” on dimensions that matter, rather than being overwhelmed with numerous dimensions right at the outset of learning. Oracle feature reduction allows the learner to assign higher weights to fewer features. This tends to improve accuracy, since the oracle selected features are the actual most predictive features. Oracle feature reduction may also improve instance selection as the learner obtains instances to query that are important for finding better weights on the features that matter. As the number of labeled instances increases, feature selection becomes less important, as the learning algorithm becomes better capable of finding the discriminating hyperplane (feature weights) on its own. This is expected given such limited training set sizes, and is consistent with most previous findings [98].

CHAPTER 6

ASKING FOR AND INCORPORATING FEATURE FEEDBACK

The experiments in Chapter 5 relied on the existence of an oracle for feature selection, either prior to the instance selection or after it. Features not selected by the oracle were dropped from the model. A human judging relevance on features at the outset requires domain knowledge and we do not want to burden the user with the task of coming up with a complete set of relevance judgments on features prior to the learning. Humans may not be able to determine relevant features in this way due to lack of knowledge or even because it is too tedious to do so. In preliminary experiments we found that users have about 60% accuracy relative to the oracle [48]. Often labeled examples may point to certain features that the user cannot gauge as relevant, as actually being relevant. For example, if the learner is able to discern from the labeled examples that *ct* is a good feature for the *earnings* category but a human does not realize this (say because she thinks *ct* stands for "Connecticut"), the learning algorithm should not zero-out the value of this feature. Therefore, the proposed system to use in practice with real users relies on a "soft labeling" of user marked features.

The above example also motivates an interleaved approach of marking features and documents. Documents provide context for the features, disambiguating them, making the feature labeling process easier for humans (In fact this idea was suggested by one of our users in the same user study). One way to implement such an interleaved approach would be to ask the user to highlight relevant features in documents that she labels. We found this approach to work quite well for a news-filtering task [88] (also discussed briefly in Section 7.6.1). However, we think that such an approach requires a much more careful reading of the document than would be needed to merely assess a document's category. Instead we opted for an approach where the user is asked to label a document and a set of features simultaneously. We choose to implement that as an interface that has a main pane with the document to be judged and a side-pane with a checklist of features to be labeled. In such an implementation, we believe, the information the learner is seeking is clearer to the human. Our tandem learning system proposed in Section 2.2, described in technical detail below, allows for such an approach.

6.1 The Tandem Learning System

The tandem learning algorithm (Figure 2.2) has two key steps that we had not elaborated on in Chapter 2: (1) *Feature_Selection* (Step 2.d.i), where a set of features for the user to label are chosen by the system, and (2) *Incorporate_Feature_Feedback* (Step 2.d.ii) where the learner incorporates the labeled features to improve its current representation of the concept. We discuss each of these respectively in the next two sections.

6.2 Asking for Feedback on Features

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

Figure 6.1. An algorithm for Interactive Feature Selection

In this section we describe our algorithm for implementing the Feature_Selection subroutine in Figure 2.2 which in this case is iterative. For a given tandem learning iteration (outer loop in Figure 2.2), the Feature_Selection subroutine queries the teacher iteratively presenting features of the set $\mathcal{P}$, in batches. The algorithm is outlined in Figure 6.1.

A budget counter B_f that keeps track of the number of features (across tandem learning iterations) that the user has been queried on is input to the Feature_Selection subroutine in addition to the current model $\mathcal{M}_t$ and the pool ($\mathcal{U}_t$). A stack of features ($\mathcal{S} = s_1, s_2, ..., s_{|\mathcal{S}|}$) to query the user on is maintained. The size of the stack $\mathcal{S}$ is dynamic as we will see. At each iteration of tandem learning the stack is initialized with the p top ranked features from the current model $\mathcal{M}_t$, such that the highest ranking feature is at the top of the stack (Step 1 in Figure 6.1). The topmost element of the stack ($top = s_1$) is popped at each iteration and the top c co-occuring features to s_1 in the pool ($\mathcal{U}_t$) are computed (Step 3.b). These c features are shown to the user. If the user marks a feature as relevant, it is pushed on top of the stack (Step 3.d). The procedure continues until the stack is empty or the budget B_f is exhausted. Typically $B_f >> c$ and $B_f >> p$. A user may be shown a minimum of 0 features in a given tandem learning iteration if the budget (B_f) is exhausted, and a maximum of B_f features (if we keep finding features greedily). Thus f in the algorithm in Figure 2.2 varies across tandem learning iterations. The algorithm is greedy in spirit: when a relevant feature is found, we keep querying the user on words that co-occur with it in the corpus, proceeding as if engaging in a depth first search on a term co-occurence graph. Our early experiments showed that there are benefits of such an approach, especially for the soft labeling method described in Section 6.3.3.

6.3 Incorporating Feature Feedback into Support Vector Machines

We now move on to describe three methods to incorporate feature feedback into SVMs (step 2.e in the algorithm in Figure 2.2).

6.3.1 Scaling

Let $\mathcal{F} = F_1 ... F_k$ be the set of features marked relevant by a user, with each F_i representing an index to a feature ($1 \leq F_i \leq N$). Let $Sc = c_1 ... c_N$ be a vector such that:

$$\begin{aligned} c_i &= a \text{ if } i \in \mathcal{F} \\ &= b \text{ otherwise} \end{aligned}$$

For each X_i in the labeled set, as well as in the unlabeled data (the pool $\mathcal{U}$ and the test set), we compute the dot product $Sc \cdot X_i$. In other words, we scale all the features that the user indicated as relevant by a and the rest of the features by b. The documents are re-normalized after scaling.

By scaling the important features by a, we are forcing the classifier to assign higher weights to these features. We demonstrate the intuition with the following example. Consider a linear SVM, $N = 2$, and 2 data points $X_1 = (1, 2)$ and $X_2 = (2, 1)$ with labels $+1$ and -1 respectively. An SVM trained on this input learns a classifier with $w = (-0.599, +0.599)$. Thus, both features are deemed equally discriminative by the learned classifier. If the feature with id 1 is indicated to be more discriminative by our user, then by our method $X'_1 = (10, 2)$ and $X'_2 = (20, 1)$ and $w' = (0.043, -0.0043)$. Thus, the feature with id 1 is assigned a much higher weight in the new model. Now, this is a softer version of the feature selection mechanisms in Figures 5.1 and 5.7. But in that case the oracle knew the ideal set of features. Those experiments may be viewed as a special case where $b = 0$. We expect that human feedback is imperfect and we do not want to zero-out potentially relevant features.

In this method the user only needs to mark whether a feature is discriminatory, and does not have to determine whether the feature is more likely to occur in a relevant document or not. It may seem that class information should be obvious to a human once she can discern feature relevance. This assumption is not true. For example, in a previous user study [48] we found that for the problem of distinguishing *baseball* documents from those on *hockey*, one of our users was able to determine that the feature *devils* was relevant; she could remember it was the name of some sport team, but could not remember whether it was a baseball team or a hockey one[1].

6.3.2 Feature Pseudo Documents

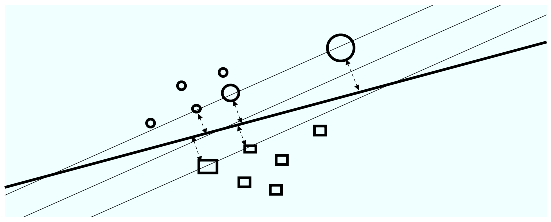
In this method we create N pseudo-documents, one corresponding to each feature, adding N more N-dimensional vectors ($\{\langle X_j, Y_j\rangle, M+1 \leq j \leq M+N\}$) to our already existing pool of M document vectors. Thus, a vector X_j corresponding to a feature-id $j-M$ will have a one in the position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

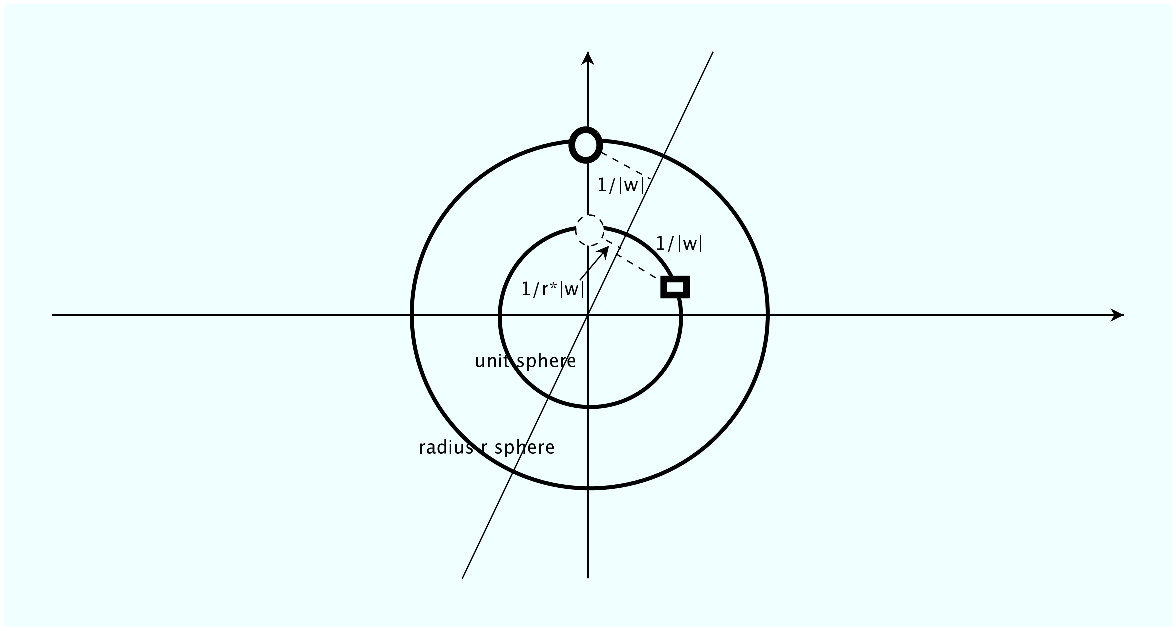
The hypothesis space is the same as before (Section 4.1.2). The version space is smaller than the original because of the added set of constraints corresponding to the feature and feature-label pairs. This method enables us to perform uncertainty sampling with the modified pool. Now the user may also be queried on *uncertain features*. An uncertain feature labeled by a user will again (in the theoretical case) result in a decrease (of the already decreased) version space by half.

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

Method 1: For support vectors that are correctly classified, the distance from the hyperplane is fixed at $1/||w||$ (refer section 4.1.3). This is forced by imposing the constraints in Equation 4.1. There are benefits to being able to control this distance for different training examples, letting more reliable instances exert a greater force on the hyperplane as shown

[1]The New Jersey *Devils* is a hockey team


(a) Weighted Margin Classifier (original idea).



(b) Feature pseudo documents and our modification of the weighted margin classifier.

Figure 6.2. Weighted Margin Classifier

in Figure 6.2(a). This idea first appeared in a paper by Wu and Srihari [115]. They modified the constraints to be:

$$Y_i(w \cdot X_i + b) \geq d_i \; \forall i = 1...t \tag{6.1}$$

Where d_i takes a value between 0 and 1, with a greater value indicating a stronger influence. Their implementation changes the optimization problem while ours is simpler.

We let the feature pseudo documents reside on an r-radius hypersphere instead of the unit hypersphere that the documents reside on. A two-dimensional example is shown in Figure 6.2(b). Let X_j be a feature pseudo document on an r-radius hypersphere. The feature pseudo-document X_j ($\forall\, j = M+1...M+N$) corresponding to a feature $j-M$ now needs to have a value r in position $j - M$ and 0 elsewhere. This support vector is forced to be at distance $1/||w||$ from the hyperplane. Projecting down to the unit hypershere (on which the documents lie), the distance is $1/(r \times ||w||)$. Hence, the feature pseudo-documents are forced to be at a distance of $1/(r \times ||w||)$ from the hyperplane on the unit hypersphere. Meanwhile the document support vectors continue to lie at distance $1/||w||$ from the hyperplane on the unit hypersphere. If $r \leq 1$ the feature pseudo document support vectors exerts a greater influence on the margin than the document support vectors. The implementation is very simple and the same QP solver used for the soft margin SVM can be used to find the solution.

The idea to use feature pseudo documents first appeared in a paper by Godbole et al [47]. Our implementation differs in that we introduce parameters to influence the extent of control of the feature pseudo-documents on the hyper-plane as compared to the extent of influence of the training documents. Note that Wu and Srihari did not use feature pseudo-documents, rather used the weighting technique in equation 6.1 to weight instances.

Method 2: The second method controls the extent of influence of the misclassified support vectors. We can do this by controlling the weight C for the feature pseudo documents by modifying the optimization problem in Equation 4.6 as follows:

$$\begin{aligned}\min_{w,b,\Xi} \phi(w,\Xi) &= \frac{1}{2}||w||^2 + C\sum_{i=1}^{t}\xi_i \\ &+\frac{C}{|\mathcal{F}|}\sum_{i=t+1}^{t+|\mathcal{F}|}\xi_i \\ \text{subject to} \quad & y_i(w\cdot x_i + b) \geq 1-\xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{6.2}$$

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

6.3.3 Pseudo Relevance Feedback

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

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

$$
\begin{aligned}
\min_{w,b,\Xi} \phi(w,\Xi) \quad &= \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{t}\xi_i \\
&\quad\; +C\sum_{i=t+1}^{t+|\mathcal{U}|} g(v_i^+)\xi_i \\
&\quad\; +C\sum_{i=1}^{t+|\mathcal{U}|} g(v_i^-)\xi_i \\
\text{subject to} \quad & \quad y_i(w\cdot x_i + b) \geq 1-\xi_i \\
& \quad \xi_i \geq 0
\end{aligned}
\tag{6.3}
$$

This idea has been used successfully in the past [115]. Like in that work, v_i is simply set to be the cosine similarity and $g(v_i) = v_i$ since v_i has the desired $(0,1]$ range.

6.4 Developing the Final Algorithm

We now develop a tandem learning algorithm using various subroutines described in the previous chapter. We use lessons learned from some initial experiments (Section 6.4.1) to lay out the task (Section 6.4.2). In Section 6.5 we discuss specifics of our implementation that may not have been enlisted previously.

6.4.1 Lessons from past work

We had conducted a preliminary user study, measuring users' abilities to mark features [48]. Annotators (volunteers) were provided with minimal information about a topic and were asked to judge a list of features, one at a time. Annotators were asked to pick between one of two choices for a given classification problem: (1) *relevant*, i.e, discriminatory and (2) *non-relevant/don't know*. They were also asked to judge a handful of documents (after judging feature relevance). Our users were of varied backgrounds and most had little understanding of machine learning techniques. The goal of that study was to determine how well naive users could identify features with little additional information that basically constituted a brief topic description. Annotators were told not to refer any other sources to make their decisions. We found that even naive users can provide effective feedback on the most relevant features (about 60% accuracy of the oracle in our experiments) resulting in performance on par with the oracle. There were many interesting results from that study. For instance, one of our annotators had difficulty in judging documents belonging to the *earnings* category in Reuters often confusing *acquisition* documents for *earnings* documents. However, she could mark 65% of the discriminatory features and generated only 0.0625% false alarms. These results can be explained by observing that the question posed to the annotator in that study was on the discriminative power of the feature. Therefore she did not have to determine whether the words *shares* was pertinent to *earnings* or not but rather she only needed to indicate whether the word was likely to be discriminatory. Additionally, one of our annotators suggested that terms shown in context would have carried more meaning. She said that she did not realize the term *ct* stood for cents until she read the documents. But since she was made to judge terms before documents she had marked the term *ct* as non-relevant/dont know. Of course in the implementation that we are planning ultimately (Figure 2.2), the user would be judging documents and terms in tandem.

We also conducted a post-labeling survey. Some of the highlights of the post-labeling survey are as follows. On average users found the ease of labeling features to be 3.8 (where 0 is most difficult and 5 is very easy) and documents 4.2. In general users with poor prior knowledge found the feature labeling process very hard. The average expertise (5=expert) was 2.4, indicating that most users felt they had little domain knowledge for the tasks they were assigned.

We also measured the manual costs of relevance feedback on features versus labeling documents: in that study [48]; we found that feature feedback takes about one fifth of the time taken by document labeling on average.

When to stop asking for labels on both features and documents and switch entirely to documents is an important question. In early experiments in this direction [48], we found that features labeled by users provided significant boosts in performance in the early stages of learning. We typically found that after 10 documents were labeled there was little use in asking for feature feedback for two reasons. Firstly, there are greater gains to doing aggressive feature selection earlier rather than later (mitigating Hughes' phenomenon). Secondly, users are able to discern certain key features only, which are often asked about in early iterations. The information about the usefulness of other features typically comes from document feedback. For example, in discerning *automobiles vs motorcycles*, the most informative words (as determined by the oracle) – *car* and *bike* – are asked of the annotator in very early iterations. The label for *car* is always (100% of the times) asked, and 70% of the time the label for this word is asked to the user in the first iteration itself. This is closely followed by the word *bike* which the user is queried about within the first 5 iterations 80% of the time. Labeling these two features in themselves results in a significant improvement in accuracy.

6.4.2 Problem definition

As with our previous experiments in Chapter 5, the task begins with a user providing two randomly labeled examples, one positive and one negative (init_size $= 2$). Then at each iteration the learner queries the user for document feedback and term feedback like in the system in Figure 2.2.

We saw that feature feedback is most useful up to the point where about 10 documents have been actively labeled. Therefore, we measure performance of tandem learning after 10 documents have been actively labeled for each topic. Note that a total of 12 documents would have been labeled at this point including the 2 original ($T = 12$). We compare the performance with traditional document-only and term-only methods. We also resorted to asking the user for 2 documents for feedback at each iteration ($I = 2$). Preliminary experiments revealed that there was little negative impact to accuracy in this approach, with some gain in efficiency.

We set the feature feedback quota, B_f, to a value of 100. Our previous research estimated that a document takes 5 times more time to label than a feature. The experiment was very conservative, and in reality we expect feature labeling to be faster, depending on the interface in which it is shown to a user. However, using this upper bound, labeling a 100 features is at most equal to the effort needed to label 20 documents. Hence we also compare tandem learning performance at $T = 12$ and $B_f = 100$ with the performance of traditional active learning after 32 documents that is, when $T = 32$ and $B_f = 0$, since labeling 12 documents +100 terms $\preceq$ 12 documents + 20 documents.

Although scaling (refer to Section 6.3.1) requires that the user only determine if a feature is discriminatory, and is probably a cognitively easier question to answer than asking the user to associate category labels to each feature (needed by the methods in Section 6.3.2 and 6.3.3), we will see that a combined approach of using all three proposed methods for feature feedback is best overall in terms of effectiveness (Section 6.4.4). Therefore in the final implementation the teacher was asked to chose one of the following options for each feature, the third choice being the default. :

1. Is the feature more likely to occur in *relevant* documents?

2. Is the feature more likely to occur in *non-relevant* documents?

3. Don't know

In the next section we use an oracle to determine which of the three feature incorporation methods is better. We find a combined approach to be best, leading us to the final algorithm outlined in Section 6.5.

6.4.3 Oracle

In the oracle experiments in Chapter 5, we cut off the ranked list at the point that yielded the highest average active learning performance (Section 5.1). However, this method was too time consuming for the larger data sets like TDT3 and RCV1 and for these experiments we cut off the ranked list at a threshold determined by the shape of the information gain curve in the following way. We took the top 30 features computed by information gain, and computed the average score. All features with scores above the average were considered "relevant" by the oracle. The oracle also associated a category label with a feature by computing the probability of occurrence of a feature in each of the two classes using all the labeled data. The feature was labeled with the category with greater likelihood of containing the feature.

6.4.4 Choice of methods

We introduced one method for asking for feature feedback and three methods for incorporating feature feedback into active learning. We conducted some preliminary experiments on the Reuters-21578 corpus to test the effectiveness of each of these methods by themselves and in combination with others.

The active learner begins with 2 randomly sampled documents, one in the positive class and one in the negative class. In choosing documents for feedback (Instance_Selection) we could use uncertainty sampling (Section 4.1.5) or random sampling. We know that uncertainty sampling works much better than random sampling for our corpora (seen from the efficiency values in Table 5.1). A third alternative is to use the top ranking documents for feedback. This is a standard approach in information retrieval and we experimented with that. We found that in general, if one is considering only document feedback, active learning works better than using top documents. A mixture of top documents and active learning, where in each iteration the user is asked to mark one top ranked document and one uncertain document is much better than just using the top ranking documents but is still not as good as asking for feedback on two uncertain documents in each iteration. However it turns out that when asking for feedback on features is included (for tandem learning that is), incorporating the top-ranking documents sometimes gives benefit, and never really performs worse than using only uncertain documents. Besides in applications like news filtering labeling the top ranked document is probably much less effort since the user is likely to read it anyways. Therefore, our tandem learner always queries the user on a mixture of top ranking and uncertain documents, whereas for the baselines we report both variants: “only uncertain documents” and “uncertain and top ranking documents”.

We conducted experiments by simulation using the oracle. Every time a feature was presented to the user for feedback, we labeled it based on the oracle’s judgment of the feature. Results of these upper bound experiments are shown in Table 6.1. Rows two and three give the performance of the two baselines. Uncertainty sampling is clearly better than using a mixture of top ranking and uncertain documents. We tuned our parameters on a handful of topics in the Reuters 21578 corpus for a system that uses all three modes of feedback together. The results of that experiment are tabulated in row 5. There is a 27% improvement in performance. We then conducted ablation experiments to study the benefit of each feature incorporation method separately (rows 6, 7 and 8) and various combinations of them (rows 9, 10 and 11). Pseudo relevance feedback (Method III) is the best performing method, and by itself gives a significant [2] improvement in performance. Scaling (Method I) on its own, also improves performance, but Method II only slightly improves performance. Method II in conjunction with each of Method I and Method III improves performance (although almost negligibly) over each of these methods individually. A combination of Methods I and III is significantly much better than the baseline, and when Method II is combined with them there is a tiny improvement over that combination. The result of using only active documents for feedback in combination with all 3 feature incorporation techniques is 0.661 (almost identical to the 0.651 value obtained using a mixture of uncertain and top ranking documents). Hence in our final implementation we use a joint approach with all three methods with a combination of active learning and topdocs. Table 6.2 shows

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

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

Table 6.1. Ablation experiments to determine which method of feature feedback is most effective. The figures in bold indicate a significant improvement in performance over the best baseline (0.516). Numbers in italics indicates a significant decrease in performance over the "uncertainty only" baseline. In all cases $T = 12$ and for the tandem learning methods $B_f = 100$.

example pseudo documents that are support vectors in the final system. The experimental setup for our final implementation that uses all three methods is described next.

6.5 Notes on the Final System: Experimental Setup, Parameters etc.

The Reuters 21578 corpus is considered to be the development set for the final system which is tested on 20 Newsgroups, TDT3 (the 1999 evaluation topics) and the Reuters RCV1 corpus. A pool size of 1000 was used for all corpora except the RCV1 corpus where the pool consisted of the training documents in the Mod-Apte split [74] (about 23K documents). The results are described and analyzed in the next chapter. The parameters p, c, a, b and r were set to values of 25, 10, 10, 1 and 10 respectively. The average performance for each topic was computed using 10 different random initializations of the initial 2 documents. We also took care that if the system had already queried the user (or

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

Table 6.2. Feature pseudo documents that are support vectors.

oracle) for a feature in a given iteration, that feature was not asked about in subsequent iterations.

6.6 Summary

We built a tandem learning system that intelligently queries a teacher on features and instances at each iteration using several intuitions and hypotheses from previous experiments. Standard machine learning algorithms do not have an easy way to incorporate feature feedback. We develop three methods of incorporating feature feedback into SVMs. The final algorithm is a fusion of ideas from machine learning, active learning in particular and information retrieval. In asking for feature feedback we picked one method that worked well. Admittedly, given the vast body of research in information retrieval on term feedback, there are a plethora of alternative methods one can explore for this component of the tandem learning system. Some of these methods may lead to better interfaces and more effective feedback. We leave such an investigation for future work (Chapter 9). In the next chapter we explore the effectiveness of our proposed methods.

CHAPTER 7

EXPERIMENTS ON A TANDEM LEARNING SYSTEM

We developed an algorithm that interactively queries a user on labels of documents and features in Section 6.4. Our experiments with an oracle in the next section verify that the learner is capable of posing questions that are most beneficial to it (hypothesis 2 in Section 1.5). We then describe our user study, verifying the hypothesis that humans can identify the features that the learning algorithm requires the answers to (hypothesis 3 in Section 1.5).

7.1 Results with an oracle

All our experiments in this section are simulations with the oracle like in Section 6.4.4 for solving the task described in Section 6.4.2. The final tandem learning system used corresponds to the one in the last row of table 6.1, described in Section 6.4. Results are tabulated in Table 7.1. We compare our results with many baselines. The "only documents" methods in rows 2, 3, 4 and 5 do not use user feature feedback at all. The second row shows the results of querying the user on one top ranked document, and one uncertain document in each round of active learning ($init\_size = 2, T = 12, I = 2$). The setup for the third row is identical to the second, except that $T = 32$. We motivated comparing tandem learning performance to this case in Section 6.4.2 because we found that the effort needed for 100 terms is at worst equal to providing feedback on 20 documents. The fourth and fifth rows show the results when both documents that the user is asked for feedback on are uncertain ones, for the cases when $T = 12$ and $T = 32$ respectively.

The sixth row shows the results of using the terms marked relevant by the oracle, and with no document feedback. For that experiment, we took all terms marked relevant by

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

Table 7.1. Results (F1) of Tandem Learning with an Oracle. Numbers in bold indicate statistically significant difference in performance over the case when $T = 32$ and $B_f = 0$ with active sampling (line 5). Tandem learning performs less than the $T = 32, B_f = 0$. Tandem learning is always better than the "only document" methods when $T = 12$.

the oracle and issued them as a query to the pool $\mathcal{U}$. A TF-IDF model was used, and the documents were ranked by similarity to the set of oracle marked terms. The top 10 documents were treated as positive documents and the bottom 10 as negative. An SVM was trained and the results on the test set are reported in the sixth row.

The seventh row corresponds to the case when the initial classifier is learned from two randomly sampled documents ($init\_size = 2$) and subsequently during active learning, the human is queried on only features. The eighth row is the result of using the complete tandem learning system. All experiments with feature feedback imposed a quota of $B_f = 100$. The effect of different values of B_f is shown in Table 7.4.

For most corpora pure uncertainty sampling is better than a combination of uncertainty sampling and topdocs when $T = 12$ (compare rows 2 and 4) and even when $T = 32$ (rows 3 and 5).

The results of learning on only the oracle terms (row six) are reasonable on most corpora, except for the TDT corpus. We wondered if our estimate of the oracle was poor for the TDT3 corpus, since the results of the sixth row may be interpreted as being reflective of the quality of the oracle. However, given the significant boost in performance for tandem learning, such an interpretation would seem contradictory. We found that the problem was due to the pool size being so much more smaller than the total collection size, that often there are not many (sometimes even zero) relevant documents in the ranking obtained using the oracle marked terms as a query. The resulting classifier learned is therefore very poor. When we increased the pool size to 10000 documents we obtained an F1 of 0.266, an acceptable number, confirming our hypothesis.

The seventh row measures performance for the scenario when after two initial documents are labeled, the active learning loop only comprises of feature feedback. Given that with 2 labeled documents the performance of the initial classifier is 0.179, 0.154, 0.053 and 0.078 for each of the four corpora respectively, we observe that we are able to recommend enough useful features to improve over the initial classifier.

Comparing the results of tandem learning (eighth row) to methods that use only documents for feedback (rows two through five) we see that for all corpora, feature feedback significantly [1] outperforms a system that uses only 12 documents for training. Tandem learning is also better than using 32 documents for feedback for three of four corpora. Remember that 32 documents is a loose upper bound on the effort required to label a 100 features. For the RCV1 corpus, labeling 32 documents results in better performance than tandem learning. However, the improvement in performance of tandem learning for RCV1 over the case when $T = 12, B_f = 0$ (row 3) is much more in magnitude than the loss in effectiveness due to expending extra effort in feature feedback instead of document feedback (compare row 8 and row 5 for RCV1), indicating that the terms labeled are quite useful, and probably only a little less useful than the documents. These results can also be interpreted from the point of view of complexity and we will use our difficulty measures to further interpret these results in the next chapter. Also note that these results depend on how good the estimate of the oracle is. Many problems in RCV1 have very few positive documents. It is possible that a better estimate of the oracle, say by using domain knowledge will boost

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

performance at least up to that of the case when $T = 32$. The results are also fairly consistent across topics as is demonstrated by Figure 7.1. 50% 25%, 86% and 98% of the topics in each of the four corpora are improved over the baseline corresponding to row 4 of the table. The topics that were improved for the Reuters and 20 Newsgroups corpora saw substantial improvements while the topics that were hurt suffered negligible change in performance.

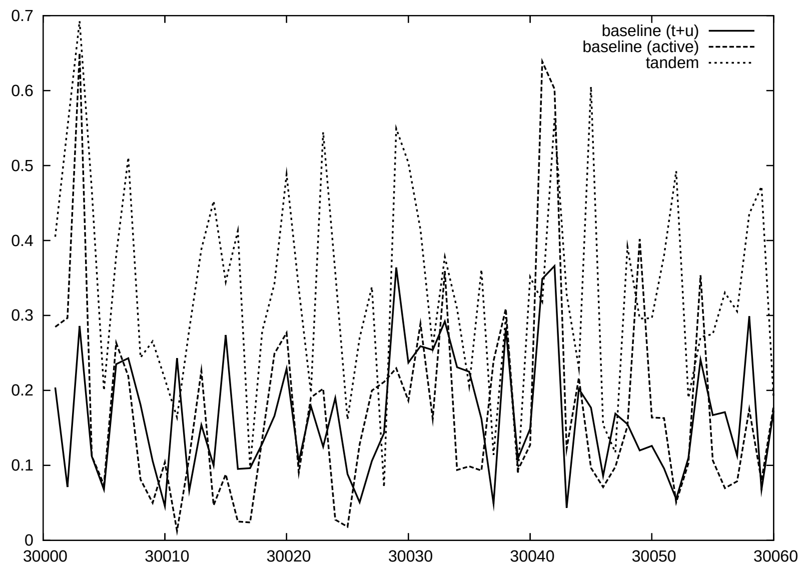


Figure 7.1. Consistency of improvements in performance across topics for classification on the TDT3 corpus. Topic ids are on the X-axis and the Y-axis is the F1 score. The baseline cases are when $T = 12$.

Such a comparison, asking how much effort is expended on marking features and whether that effort is better spent in marking documents, differentiates our work from all past work in machine learning that uses user prior knowledge to boot-strap learning [115, 47, 35]. From our observations, spending some effort to mark features is almost never an effort wasted. Many times it boosts learning, improving over a paradigm that just uses documents for feedback, and does not hurt performance. Features are also not sufficient in themselves (compare row 8 with rows 6 and 7).

Hypothesis 2

The results with the oracle prove that tandem learning is effective, showing that our method for asking features and incorporating feature feedback indeed works quite well. We can thus check off hypothesis 2 in Section 1.5 as proven. The experiments reaffirm our belief in hypothesis 1 (that there exist a set of features which if labeled, can bootstrap learning) which continues to remain true even with the soft-labeling approaches developed in Chapter 6.

7.2 Results with Real Users

We now ask the following questions:

1. Can humans label features as well as documents? In other words, are features that are important to the classifier perceptible to a human?

2. If the feature labels people provide are imperfect, is the feedback still beneficial to active learning?

We obtain feedback on features offline using a TREC "pooling-like" approach discussed in Section 7.2.1. We then use the judgments so obtained to measure the effectiveness of user marked features on the tandem learning system in the same way in which we simulated a human-in-the-loop with the oracle. We employed this methodology rather than one where an actual user labels features and documents in tandem because our approach is cheaper and allows us to run many repeated trials of our experiments, also enabling us to do significance testing. We reserve a more realistic evaluation with a true human in the loop for future work.

7.2.1 Experimental Setup

We ran the system described in Section 6.4 for 10 different choices of the initial positive and negative documents (Step 1 in Figure 2.2) for the 60 topics in the TDT3 corpus. Let us call each initialization for a given topic a “run”. We concatenated all recommended terms (Step 3.a) keeping track of how many times a term was recommended across different runs (remember that a term is not recommended more than once for a given run). We discarded all terms that were recommended only once across all runs for a topic. For each topic we then added the terms determined relevant by the oracle into the pool giving us an average of 130 terms per topic. We now proceed to describe how we obtained judgments on terms for each topic.

We had one paid annotator judge 60 topics in the TDT3 corpus. She was not a computer scientist, rather a political science major, computer literate and familiar with some basic statistics. She was briefly explained the task in a 15 minute training session. She was also given written instructions as documented in Appendix B.1. She was given a brief description with access (as a hyper-link) to a detailed topic description before she began making terms. Both the brief and detailed topic descriptions are provided by the LDC.

We sorted all terms alphabetically, and showed her each term with 4 contexts in which it appears in the corpus. A small pane with a reminder of the topic (the brief topic description) and a link to the more detailed description was available to the user at all times. Screenshots of the interface are available in Appendix B.2. We retrieved context for each feature by issuing that term as a one word query using the Indri toolkit [104]. Context helps disambiguate a word and also provides a snapshot of the senses it appears in, in the corpus. Other than our standard example of context helping a user determine the meaning of the word like *ct*, another example where context proves to be particularly important is in having the human overcome the effects of machine translation and ASR errors. Figure B.2 shows a screenshot where the user is asked to judge the term “bell” for the topic *Nobel Prizes are Awarded*. The word “nobel” is consistently mis-translated as “promises bell” in machine translation documents. A user might be able to notice such an error and mark the word “bell” as associated with the relevant documents.

Terms were shown one at a time and at each instance the user was asked to mark one of the three choices in Section 6.4.2. We imagine that in a more realistic implementation, terms will be shown as lists, which is probably faster than having a user judge one feature at a time. The term-at-a-time method is however, the best interface for a controlled experiment to measure users' abilities to mark features. In fact in an earlier study we did not even show context and the user was given a very brief topic description. One option for this study was to show an initial screen with a list of features, and for each feature a hyper-link to a context, in case a user needed clarification about the meaning of a feature. We avoided this interface because it is possible that a user might not click on a link because of a pre-conceived notion of what the feature means. For example, a user who assumed *ct* stood for Connecticut and never imagined it could mean anything else (cents in our case) would not click on the link.

The author of this thesis also judged terms for all 60 topics. Although one might think that the author would represent a biased user, with domain knowledge of the corpus, the underlying algorithm and so on, surprisingly it turns out that she and the paid annotator perform almost on par, especially in terms of the final effectiveness of the tandem learning algorithm (Section 7.2.2). The author admittedly was not as careful as User 1 (the paid annotator) was. She only read the brief description and did not change the default "don't know" unless she was absolutely sure about the relevance of a feature. User 1 took a median of 3, 3 and 2 seconds to mark terms associated with the relevant, non-relevant and don't know classes respectively. User 2 (the author) on the other hand took 4, 3 and 1 seconds to mark features in each of these 3 classes.

There are pros and cons of this offline approach of obtaining features. The main advantage was that it was cheaper to obtain features this way rather than using an online approach where the user evaluated the real system. We used the LDC judgments on documents, and reserved our annotators efforts only for feature judgments. The annotator did not have to judge a given feature for a given problem more than once. We also discarded features that were rarely asked by any algorithm, saving annotator effort significantly. We could repeat experiments as simulations with our database of relevance judgments allowing us to further develop algorithms for interactive feature selection, testing them using a simulation-like approach.

The cons of such an approach are that features that are not judged, either because they were discarded or because the system being evaluated was not in the pool will never be judged as relevant. By tossing in the oracle features into the pool of features on which we obtained judgments, we hope that we have obtained judgments on all key features, and the ones that we miss are probably not as critical. Our offline method also does not capture the true effects of a user judging terms and documents in tandem; the growing knowledge of the user as she reads more documents in the corpus; effects of boredom and so on. Rather, our paid annotator dedicatedly spent time marking features. In fact she was given the option to take breaks between annotations. Nevertheless, what the user study captures is that a human can indeed judge relevance of features to the extent that results in an improvement in performance almost equivalent to that of the oracle (Section 7.2.2). Even without paid annotators, in a previous study conducted with volunteers of different backgrounds [48], we found that users could identify useful features sufficiently well with minimal knowledge. We provided some more knowledge of the topic and designed a better interface for this user study based on results from the previous one (refer Section 6.4.1).

7.2.2 Results

We now describe the results of our user study first comparing the user intrinsically to the oracle and then measuring the effect of the user labeled features on tandem learning performance.

Inter-annotator agreement:

We measure the extent to which our users tend to agree with each other about the importance of features using the kappa statistic [25], a measure that quantifies the agreement between annotators that independently classify a set of entities (in our case the features) into classes (relevant versus non-relevant versus don't know). Kappa is given by:

$$\text{kappa} = (p_o - p_c)/(1 - p_c) \tag{7.1}$$

Where p_o is the observed proportion of agreement and p_c is the agreement due to chance [25, 67]. Table 7.2 shows the kappa values for each of the two users for the 3-way classification problem (columns 2 and 3).

Landis and Koch [67] provide a table giving guidelines about how to interpret kappa values. This table is given in Appendix D. According to their table, the agreement between User 1 and the oracle is "poor", but the agreements between User 1 and User 2 and between User 2 and the oracle are "fair". Upon investigation we found that User 1 had a tendency to attribute many features to the "non-relevant" category. User 2, on the other hand typically marked only "relevant" features, leaving all others to the default "don't know" category. This tends to match with the oracle to quite an extent. The oracle marked 12 features (on average over the 60 topics) as belonging to the "relevant" class and 0.013 features as belonging to the "non-relevant". The negative class in one-versus-all problems is arguably harder to model statistically [79] and the oracle captures this effect. Such domain knowledge may have biased User 2, who is the author of this thesis and has higher agreement with the oracle. If we collapse, the "non-relevant" and "don't know" categories into one, giving a 2-way classification problem (columns 4 and 5), we see increased agreement between the two users, reflecting an overall "moderate" (and bordering on "substantial") agreement. In fact in our preliminary user study [48] with 5 users we found an agreement of 0.68. In that study users were strictly asked about the "discriminatory" power of a feature.

We also measured precision and recall of each of the users with respect to the oracle. This is also tabulated in Table 7.2. As mentioned earlier, the oracle had extracted about 12 terms on average per topic. Users tend to be more verbose than the oracle, with User 1 judging 25 terms (average) in the "relevant class", 29 terms in the "non-relevant" class, and the remaining terms in the "don't know" class. These numbers are 15, and 0.33 respectively for User 2. Both users have very high recall, but relatively lower numbers of precision. Appendix C shows the terms marked by the oracle, as well as the positive and negative terms marked by both users for three example topics: *Osama Bin Laden Indictments, Nobel Peace Prize Awarded* and *Taipei Mayoral Race*. Remember that the oracle is constructed from a feature selection algorithm, which might suppress redundant features, whereas the users did not do so. Ultimately it should be the recall with respect to the oracle that matters for effectiveness. In fact it is indeed the case that performance of the algorithm using user labeled features is almost on par with that of the oracle. This is seen from the last two columns of Table 7.2.

Hypothesis 3

In experiments in this section we have shown that users are capable of selecting the key features necessary to bootstrap active learning, hence proving hypothesis 3 (Section 1.5). Although users may mark more features than are necessary or miss a few features occasionally, the ultimate performance achieved by using user labeled features, compares with the performance obtained using the oracle.

7.3 Performance on ranking metrics

We also measured performance with metrics that measure ranking rather than classification accuracy. We do this for better interpretability of results, to understand where one algorithm differs from the other. We measured mean average precision (MAP) and precision at rank 5 (P@5) (Table 7.3) . Average precision (AP) is the average of the precision values computed at each point in the ranked list where a relevant document is retrieved. Mean average precision is the average of the AP scores over topics.

Tandem learning results in a significant improvement of MAP, over the baseline case of $T = 12$, for three of four corpora. Only 20 Newsgroups sees a significant improve-

User	Ability to mark features								Effectiveness	
	Kappa				P & R				F1	
	3 class		2 class		P+	P-	R+	R-		
	User 2	Oracle	User 2	Oracle					2 class	3 class
User 1	0.275	0.147	0.569	0.305	0.402	0.000	0.789	0.900	0.316	0.297
User 2	-	0.350	-	0.359	0.565	0.883	0.649	0.900	0.286	0.287

Table 7.2. Inter-annotator agreement and performance using the user labeled features. P+ and R+ denote the precision and recall of the features the user labeled in the “relevant” class with respect to the features that the oracle ascribed to the “relevant class”. P- and R- denote the corresponding numbers for the “non-relevant” class. Performance (F1) of tandem learning using user labeled features is comparable to that of the oracle performance of 0.336, and is significantly better than the baseline of 0.176 (corresponding to row 2 of Table 7.1). The 0.316 performance obtained using User 1’s positively labeled features is statistically indistinguishable from the performance of the oracle.

Metric		Reuters	20NG	TDT3	RCV1
P@5	Baseline (12 docs)	0.828	0.748	0.802	0.584
	Baseline (32 docs)	0.815	0.640	0.828	0.676
	Tandem	0.85	**0.844**	0.797	0.640
MAP	Baseline (12 docs)	0.433	0.238	0.480	0.223
	Baseline (32 docs)	0.679	0.397	0.546	0.380
	Tandem	**0.613**	**0.393**	0.481	**0.292**

Table 7.3. Effect of tandem learning on Precision at 5, and on mean average precision (MAP). Numbers in bold are statistically significant improvements over the baseline with 12 documents.

ment in P@5. The TDT3 corpus experiences a drop in both MAP (insignificant) and P@5(significant) using tandem learning. Tandem learning seems to be improving recall, sometimes at the expense of precision.

It is intuitive and widely accepted that methods that are tuned for classification accuracy are not optimized for MAP [71, 58]. A system tuned for mean average precision will score better if it gets more relevant documents at the top of the ranked list. A similar intuition applies for P@5. A measure of classification accuracy like F1 does not give a system a better score for improving the top of the ranked list, rather it concentrates on the ability of the system to discriminate documents, focusing more on the boundary of separation between the two classes. These results may be altogether different had a different method, optimized for ranking metrics, been used. Nevertheless, we are happy to see an improvement in MAP using tandem learning. There has been some recent work in training SVMs differently for different performance metrics [58, 23], and we plan to experiment with these new techniques in the future. The objective of tabulating these observations was to understand where feature feedback was impacting our methods.

7.4 Varying B_f

We wondered how performance would be affected for different values of the feature labeling budget, B_f. Table 7.4 shows the F1 scores for different values of B_f for the

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

Table 7.4. F1 for different feature feedback quotas. In all cases $T = 12$. Numbers in bold are statistically significant over the previous row.

Reuters and 20 Newsgroups corpora. For the Reuters corpus, even a small value of B_f, like 10, results in a big improvement in performance over the case when $B_f = 0$. We see less value in increasing the budget beyond a B_f value of 40. The results for the 20 Newsgroups data set are similar except that F1 increases quite steadily till $B_f = 80$. In the next chapter we will find that 20 Newsgroups is of higher feature complexity than Reuters and this may be a possible explanation for requiring more feature feedback for 20 Newsgroups than for Reuters.

7.5 An online task

System	T=4	T=6	T=8	T=10	T=12
Baseline	0.199	0.297	0.345	0.357	0.346
Tandem (Oracle)	**0.271**	0.291	0.345	0.379	0.362
Tandem (avg)	**0.301**	**0.340**	**0.389**	**0.408**	**0.386**

Table 7.5. Performance on a news filtering task for different values of T. Numbers in bold indicate statistically significant improvements in performance over the baseline.

In the official TDT tracking evaluation [3], the system is given one training document per topic. The test data for each topic consists of a stream of documents that arrive in chronological order and need to be declared as on or off the topic of the training story. The task is online and the system is expected to process the stream in order and no look

T	Metric	F1	MAP	P@5
$T = 4$	Baseline	0.199	0.480	0.360
	Tandem (Oracle)	**0.271**	0.500	0.355
	Tandem (avg)	**0.301**	**0.550**	**0.396**
$T = 12$	Baseline	0.345	0.571	0.304
	Tandem (Oracle)	0.362	0.559	0.305
	Tandem (avg)	**0.386**	**0.630**	**0.330**

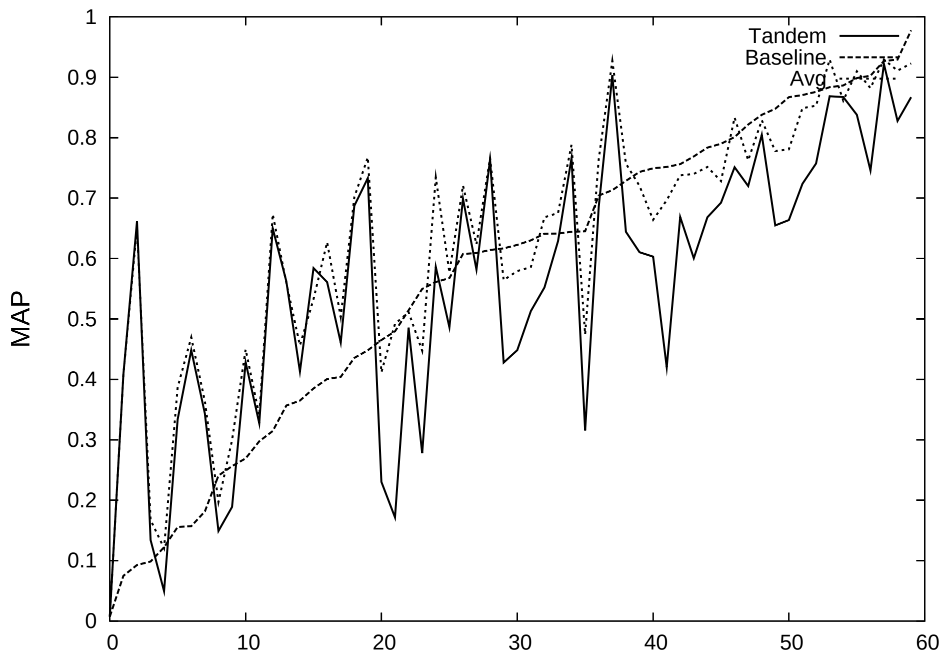
Table 7.6. Performance on a news filtering task. Numbers in bold indicate statistically significant improvements in performance over the baseline.

ahead is allowed. In the unsupervised tracking task no feedback is allowed after the initial training document is provided. In the supervised adaptation track the user provides a relevance judgment on every delivered document. The two tasks represent two extremes of an interaction spectrum. Leuski and Allan [70] studied a more realistic version of the TDT task, wherein the system is evaluated at regular intervals (a day, half a day, a week and so on). Just like their work, we modified the news filtering scenario of TDT to a more realistic one. In the current task, the user marks the first relevant document on a topic as relevant. The system also picks an arbitrary “off-topic” document with a time-stamp not greater than that of the relevant document. Filtering then begins. Interaction now happens after batches of 500 documents (roughly a day’s worth of documents) arrive. The user is queried on one top ranking document, and one uncertain document, both sampled from the current batch. Feature feedback is performed at this time using the same algorithm as described in Section 6.5. Feedback in this case also continues only for 5 iterations.

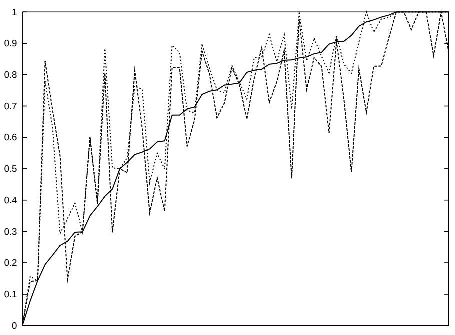
SVMs have performed quite well in the TREC filtering tasks, especially on evaluation on the F-Measure and MAP for the batch filtering and routing tasks [73, 22, 100]. Tandem learning has maximum impact (statistically significant) when $T = 4$. This is seen from Table 7.5 where we report F1 for different values of T. Table 7.6 shows performance on F1, MAP and P@5 for the baseline system and the tandem learning system for two values of T. Many of the batches have only one relevant document, which if missed gives an F1 score of 0 to the system on that buffer. Therefore, for further analysis we focus on MAP.

On close examination we found that tandem learning and the baseline perform differently for different queries. Figure 7.2(a) shows the performance (MAP) of the baseline system and the tandem learning system ($T = 12$ for both systems) for the 60 topics, sorted in order of performance of the baseline. Tandem learning improves performance on very poorly performing topics, but decreases performance on some topics for which the baseline performance is very high. The standard deviations of the baseline system and the tandem learning system are 0.26 and 0.22. By combining the two systems by simply averaging their scores we get performance corresponding to the curve 'Avg'. The resulting system's performance is also shown in the last line of Table 7.5 (standard deviation on MAP when $T = 12$ is 0.22) and the fourth and seventh lines of Table 7.6 . We tested the performance of the three systems (in Table 7.5) on the topics used in the evaluation in years 2000, 2002 and 2003. The performance is similar as seen in Figures 7.2(b) and 7.2(c). In all cases the tandem learning system performs better than a system with $T = 32$ documents (and no terms) for feedback. For the topics used in the evaluations in years 2000, 2002 and 2003, many topics have only a handful of relevant documents and hence it is easy to see why increasing the document feedback quota to $T = 32$ does not improve performance significantly over the case when $T = 12$. We observed a similar such result with our passage filtering system, where we found that combining the scores of two systems, one with document feedback and one with passage feedback, results in overall improvements in performance and that document feedback saturates pretty quickly [88].

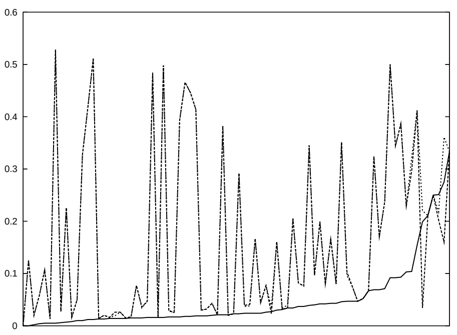
A point worth mentioning here is that the pool for this task is not a random sample of the data. The fact that this can cause many problems for active learning [83] leading to over-fitting of the classifier, is a possible explanation for why the average classifier works better than either the purely document based one or the tandem learning one. It is possible that similar or better results may be obtained if the parameters of the system were tuned for



(a) Performance on the 60 1999 Evaluation topics on the TDT3 corpus. MAP scores are 0.571, 0.559 and 0.630 for the Baseline Tandem and Avg systems ($T = 12$ and $B_f = 100$). Performance using only documents at $T = 32$ is 0.659.



(b) Performance on the 60 2000 evaluation topics on the TDT3 corpus. MAP scores are 0.693, 0.678 and 0.741 for the Baseline, Tandem and Avg systems ($T = 12$ and $B_f = 100$). Performance using only documents at $T = 32$ is 0.697.



(c) Performance on the 80 2002 and 2003 evaluation topics on the TDT4 corpus. MAP scores are 0.048, 0.165 and 0.170 for the Baseline, Tandem and Avg systems ($T = 12$ and $B_f = 100$). Performance using only documents at $T = 32$ is 0.048.

Figure 7.2. News filtering performance. All graphs are sorted by the baseline performance. Notice that tandem learning helps boost the scores of the poorly performing topics, sometimes at the expense of hurting performance for which the baseline system performs very well.

filtering. However, we wanted to demonstrate the easy portability of our classifier from an offline to an online scenario.

In our previous work in Topic Detection and Tracking we had studied the evolution of important keywords in news topics [88]. Towards this goal we defined a measure of informativeness of a document. We compute a list of information gain scores for all terms in the vocabulary of the corpus for each topic. The most informative keywords would have the highest score. Then for each topic, we order the documents by their time of appearance in the news. Let n be an index on the time of the document such that the n^{th} document appears after the $(n-1)^{th}$ document. Then proceeding in the order of time, we sum the information gain scores of terms that first appear in a document n. In this way, each document gets an *informativeness score*. Note that if a term has already occurred in a previous document on that topic, it does not contribute to the informativeness score. The informativeness score measures how many new important words appear in each document for a given topic. We can thus plot the informativeness scores over time for each topic. Since information gain scores are not normalized, we normalize the informativeness score for each topic by dividing by the maximum for that topic. A normalized informativeness score of 1 then represents the most informative document for that topic. Figure 7.3 shows the normalized informativeness score of the n^{th} document on a topic (sorted by the time at which the news story appeared) for all 60 topics (1999 evaluation) in the TDT3 corpus. From the plot in Figure 7.3 it seems like the key informative terms appear in the first few documents. If a user is able to mark the important terms fairly early, much of the information needed to learn the topic is captured. Thus a tandem learning approach works well for news filtering.

7.6 Additional Experiments

We now briefly describe other systems that we have constructed that use alternate forms of feedback other than document feedback to bootstrap learning.

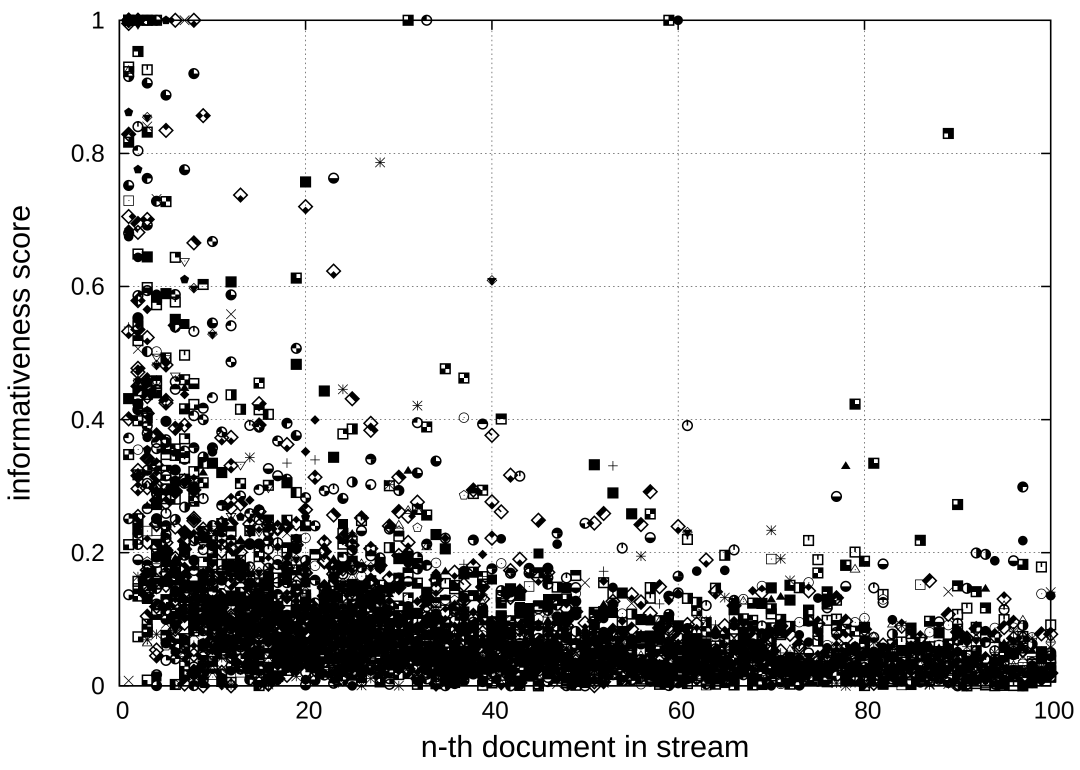


Figure 7.3. The evolution of terms with time: The informativeness scores of the n^{th} document on a topic. The x-axis is the n^{th} document. The y-axis is the informativeness score. The thick line traces the average informativeness score over all topics. Most of the informative terms appear in the first 20 documents.

7.6.1 Other Forms of User Input: Forms and Passage Feedback

In our previous work we used the topic descriptions as provided by the LDC (See Appendix C for examples) to obtain a list of user marked features for feedback [48]. The topic descriptions are structured and we used words in the *who*, *what*, *where* and *when* fields to obtain a list of relevant features. We used these features for tandem learning (using the scaling method only). The scenario is equivalent to asking a user to fill a form containing some structured questions about the topic. These questions would depend on the domain. In this case, the domain being news, we know that people, places and organizations are important. Bearing in mind that the TDT annotators are experts in assessing the topicality of news, we observed that we could use the information from the topic description to obtain

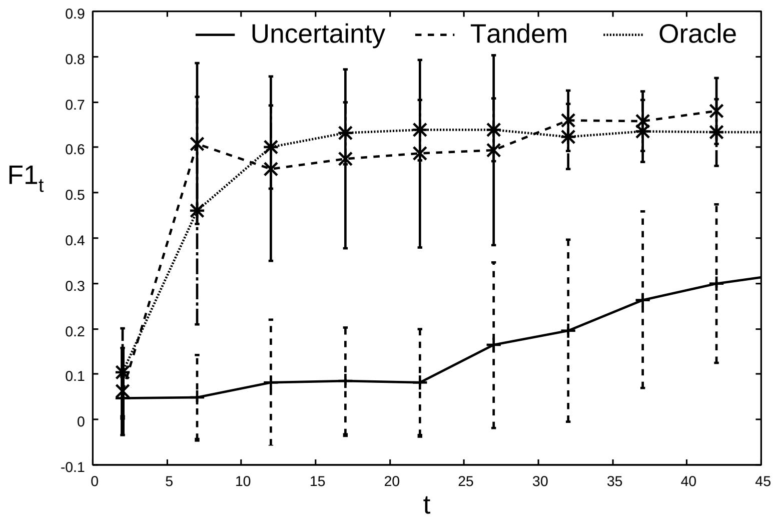


Figure 7.4. Learning curves for *Hurricane Mitch* with the x-axis being the number of labeled documents and y-axis $F1$

performance on par with the oracle. An example output from that work, for the topic of *Hurricane Mitch* is shown in Figure 7.4.

In yet another piece of work [88], for the supervised filtering task in the Topic Detection and Tracking evaluations, we asked users to highlight relevant passages of text as and when they read documents in order to assess them for relevance. The system was built on a Rocchio classifier, with one classifier built using the document level judgments and another built using the passage judgments. The scores were combined by linear interpolation. The resulting classifier showed significant improvements in performance on both the TDT and TREC filtering metrics.

7.6.2 User Prior Knowledge for Clustering

In work with Bekkerman et al [15], we built a system that can cluster by any user specified criterion. The system leverages user prior knowledge on features to understand the kinds of clusters that a user is interested in. For example, if the user is interested in clusters by sentiment for movie reviews, she may specify the keywords *good* and *terrific* for one cluster and *terrible* as an example feature for the other cluster. If the user cannot specify features, for example, in classifying by genre, the system defaults to clustering by the underlying document representation. We found significant improvements over the baseline for sentiment classification using our method and found that naive users were able to mark features sufficiently accurately even from prior knowledge alone.

7.7 Summary

In this Chapter we found that our proposed algorithm for tandem learning using an oracle typically gives significant improvements in performance over traditional active learning. We also showed that such a tandem approach results in significantly greater improvements over using only the oracle chosen terms. We found that users have sufficient enough recall as compared to the oracle to result in performance comparable to that of the oracle. We applied our tandem learning approach directly to an on-line task and found that tandem learning improves performance over using only documents for feedback. Our user studies have pointed to various (anecdotal) examples of when and why they can mark useful features, leading to several ideas for a more thorough user study in the future.

CHAPTER 8
THE COMPLEXITY OF CONCEPTS IN TEXT CLASSIFICATION

In the real world some concepts are easier to grasp than others. As pointed out in the introduction, the concept of a “bird” is probably much easier and quicker to learn than the concept “art”. 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 will refer the interested reader to the works of Feldman [43], Chater [85] and others [102]. 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 concept[1]. He found a surprising empirical ‘law’: the subjective difficulty of a concept in human learning is directly proportional to the Boolean complexity of the concept. In this chapter we wonder about the variance 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.

[1]Example boolean expressions for a text classification task are shown in Table 1.1

Given a learning algorithm, a set of features and a concept, there is some maximum achievable 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. Given such a set of concepts that are "almost linearly separable", that is "learnable" by a linear SVM, we question how much training is "adequate" to attain the maximum achievable accuracy.

One view of concept complexity or difficulty may be one associated with 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 chapter we restrict ourselves to concepts that we know are ultimately learnable by the chosen algorithm (SVMs) and ask how easily 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 than others. Whereas 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 begin by defining a set of difficulty measures in Section 8.2 based on the number of training examples and the number of features needed to achieve the maximum accuracy for the learner. Our instance complexity measures intuitively capture the number of training examples needed to attain the maximum achievable accuracy. These examples need not be random; they can be intelligently picked. We aim to capture the minimum number of training examples needed to learn a concept. 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 which captures the minimum number of intelligently picked features needed to achieve the maximum possible accuracy. If a concept can be described by a weighted combination of a few well selected features it is considered to be of low complexity. We find a high positive correlation between instance complexity and feature complexity (Section 8.4.1) using our measures. We then try to put together what these results mean for text classification, machine learning and tandem learning in Sections 8.4.3, 8.5, 8.6 and 8.7.

8.1 Data

Before we proceed further we describe the data and the kinds of concepts we are studying in this chapter. As usual, we are only concerned with text categorization problems. We consider 9 corpora and 358 binary classification problems as shown in Table 8.1, 4 of which were already introduced in Section 4.2. In computing complexity for the Reuters-RCV1 corpus we only used the 23149 training documents from the Mod-Apte split [74] for efficiency purposes.

Most corpora have topic-based category labels, except for three: (1) the Topic Detection and Tracking corpus that contains classes based on events (Section 4.2) (2) the British National Corpus BNC corpus where the classes are based on genre. (3) The documents in the Enron corpus are email 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.

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 8.1 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.

8.2 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

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 8.1. 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.

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 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 trained with all features and ample training examples. In computing instance complexity we use active learning methods which give us an experimental upper bound on complexity. The tightness of the bound is dependent on the active learning method used.

Similarly, our feature complexity measures quantify 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.

8.3 Instance Complexity Measures

Given a classification algorithm and a binary classification problem, there is some maximum achievable performance ($F1_M(\cdot, N)$), often under 100% in practice (Table 8.1). Refer Section 5.2 for notation. 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 would then be computed as $\hat{i} = min\{\operatorname{argmax}_i F1_i(\cdot, N), i\text{=1...}2^M\}$.

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.

As before (Figure 2.1) the active learning algorithm 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, $F1_{2^t}(ACT, N)$ of the classifier after every 2^t iterations of active learning with t varying as $1, 2, ..., \log_2 M$, where M is the total number of instances available for training[2] . A performance curve for three problems in the 20 Newsgroups data set is shown in Figure 8.1. 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 feature 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

[2]Note that the use of t is slightly different in this chapter, denoting the log of the number of training examples, and not the actual number of training examples.

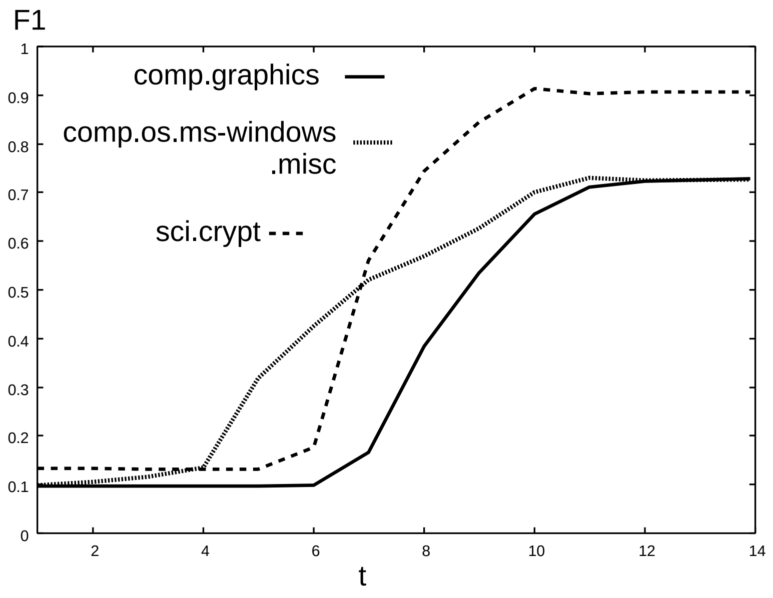


Figure 8.1. Learning curves for a single classifier on 3 problems.

less complex than *comp.graphics*. The approximate complexity value of 2048 estimated using active learning does not capture this learning rate. We factor in the learning rate by considering the area under the learning curve (as we did with the efficiency metric in Section 4.3.2) computed as :

$$\text{AUC}_{\text{log}} = \sum_{t=1}^{log_2 M} F1_{2^t}(ACT, N)$$

This time we measure performance at exponentially increasing intervals, and compute the area under the learning curve, plotted with a logarithmic X-axis. AUC$_{\text{log}}$ 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, independent of M, we define the *active learning convergence profile* as follows:

$$p_{\text{al}} = \frac{\sum_{t=1}^{log_2 M} F1_{2^t}(ACT, N)}{log_2 M \times F1_M(ACT, N)} \qquad (8.1)$$

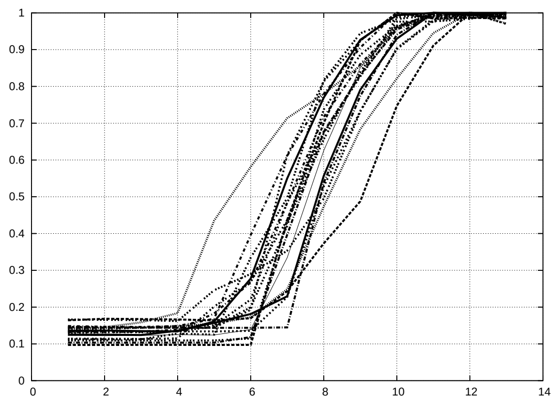
p_{al} is the area under the normalized active learning curve (See Figure 8.2(a)), with a range between 0 and 1 and is independent of M. Higher p_{al} implies faster convergence. The p_{al} values for the three problems in Figure 8.1 – *ms-windows.misc, comp.graphics* and *sci.crypt* are 0.61, 0.45 and 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 $I_{pc} = 1 - p_{\text{al}}$. 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 logarithm of the number of instances needed to achieve 95% of the best performance. We expect that n_i is an upper bound on $log(\hat{i})$. 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 of improvement at the final stages of learning, before the concept is fully learned, is very slow with several thousands of instances contributing to a tiny improvement in performance, unnecessarily inflating the complexity score (See Figure 8.2(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.


(a) Active Learning

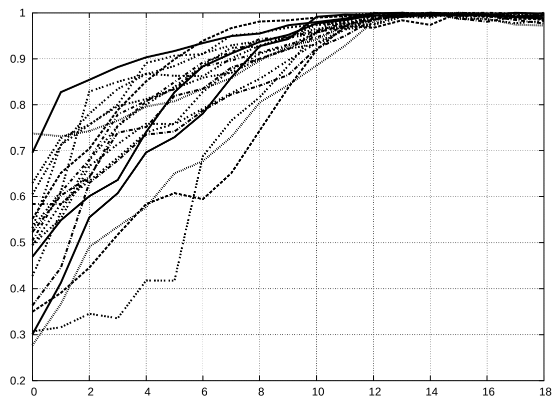

(b) Feature Learning

Figure 8.2. Normalized learning curves (active learning and feature learning) for 20 Newsgroups.

8.3.1 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. Again, instead of evaluating 2^N combinations of features, we estimate an approximation of the true feature complexity by using an oracle (Section 5.1) 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 2^k top ranking features where k varies between 1 to $log_2 N$. The normalized area under this feature learning curve, *the feature learning convergence profile*, p_{fl} is computed as follows[3] :

$$p_{\text{fl}} = \frac{\sum_{k=1}^{log_2 N} F1_M(ACT, 2^k)}{log_2 N \times F1_M(ACT, N)} \tag{8.2}$$

[3]Note that the use of k is slightly different in this chapter, denoting the log of the number of features, and not the actual number of features.

Normalized feature learning curves for the 20 Newsgroups corpus are shown in Figure 8.2(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 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.

Methods

One method for instance selection is SVM uncertainty sampling [75] as discussed in Section 4.1.5. However, this would involve retraining the SVM O(M) times, which can be very time consuming. 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 a another active learning method **– a committee of perceptrons** [33]. The perceptron being mistake-driven needs to be retrained fewer than M times (M times in the worst case. See Section 4.1.1). Besides, each retraining unlike the SVM, does not involve solving a quadratic programming optimization problem (See Section 4.1.3). Of course, active learning using perceptrons may not be as effective as SVM uncertainty sampling. However, we find that the ranking problems by their complexity computed using the perceptron committee is almost identical to the ranking obtained using SVM uncertainty sampling (refer Figure 8.6), though the actual values of complexity obtained with SVMs is probably a better estimate of the true complexity.

We can use information gain as described in Section 5.1 for feature selection. However, information gain does not ignore redundant features. So we also experimented with **SVM**

LARS [61], 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.

Each time we compute $F1_{2^t}(ACT, 2^k)$ in equations 8.1 and 8.2, 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 (refer Equation 4.1). 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 8.1, the last column lists the Max F1 values obtained with a 90-10 training-test split of the corpus.

8.4 Results

We describe the results of using our feature complexity measures on the 358 problems described in Table 8.1.

8.4.1 Correlation of Instance Complexity and Feature Complexity

Figure 8.3 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

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

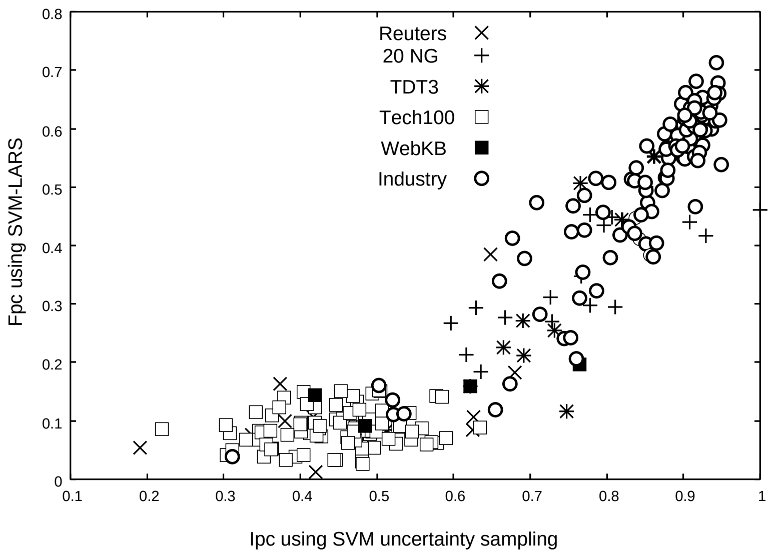


Figure 8.3. Correlation between I_{pc} and F_{pc} using SVM and LARS. Correlation of instance complexity and feature complexity is independent of methods used to compute the two.

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. Table 8.2 below shows the correlation coefficients for I_{pc} and F_{pc} for various combinations of classifiers, instance selection mechanisms and feature selection mechanisms for the 6 corpora in Figure 8.3.

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

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**

Table 8.2. I_{pc} and F_{pc} for various combinations of classifiers, instance selection mechanisms and feature selection mechanisms. Numbers in bold indicate statistically significant correlations at the 95% level of confidence.

complexity[5] of the learning algorithm. That our complexity measures exhibit this correlation substantiates our belief in these measures.

8.4.2 Difficulty of Domains

We now benchmark all 9 corpora as easy or difficult for active learning using our complexity measures. Table 8.3 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 8.5. 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 8.3. 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.

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 8.4 (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

[5]The complexity of an algorithm 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.

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 8.3. 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 & info. gain for feature selection.

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 8.6. 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* [14, 56].

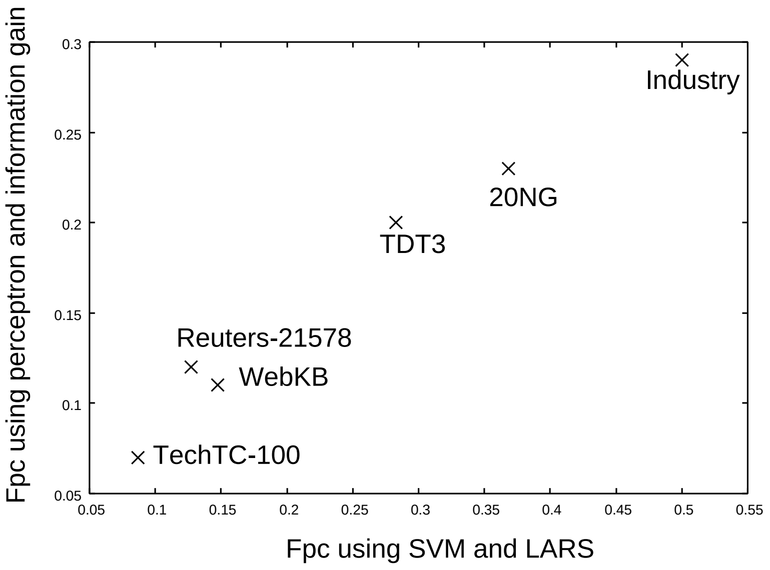


Figure 8.4. Ranking using F_{pc} computed by two different methods results in a similar ranking of corpora.

The Tech100 data set is a result of the efforts of Davidov et al [34] 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 8.4.

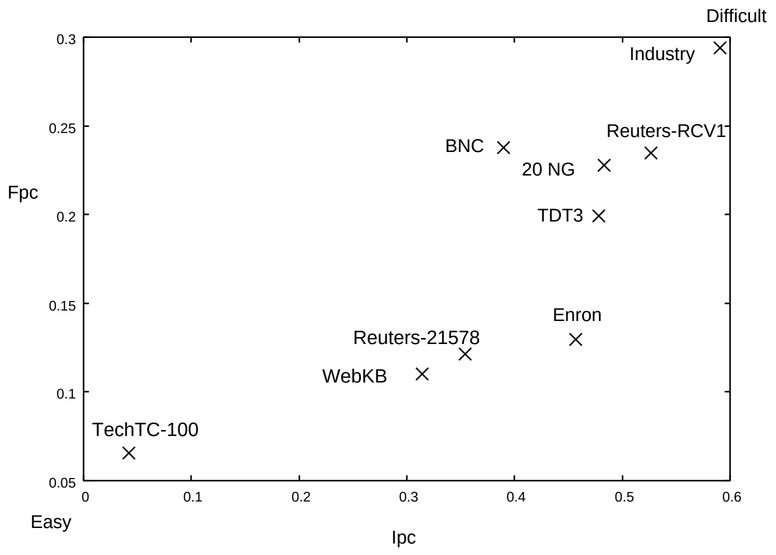


Figure 8.5. Instance Complexity (I_{pc}) and Feature Complexity (F_{pc}). A higher value of complexity indicates a difficult problem. Notice how instance complexity and feature complexity are correlated.

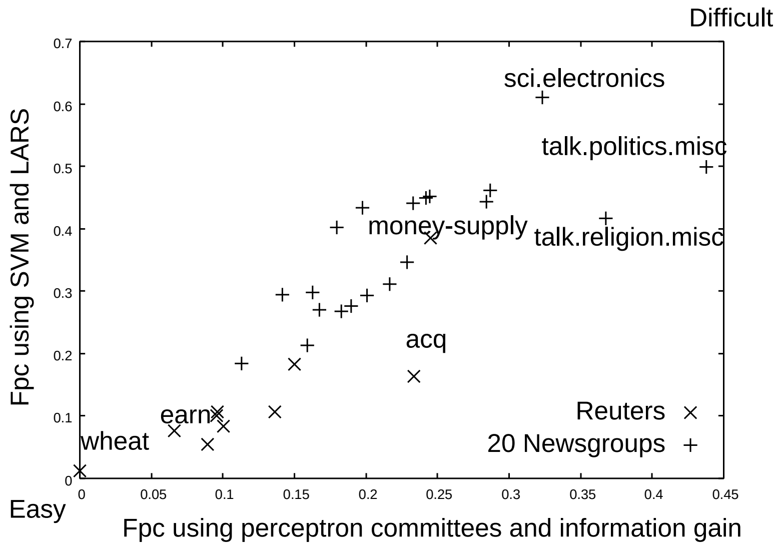


Figure 8.6. Feature complexity (F_{pc}) scores of problems in the Reuters-21578 and 20 Newsgroups corpora computed using 2 different methods. Higher the complexity more difficult the problem.

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 important 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.

	C_f by class type			
Subset of TDT3	Events	*Nobel Awarded*	Subject	*Legal & Cri- -minal cases*
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 8.4. 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 8.4. The fifth column shows the C_f values for

[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*

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 fourth column of Table 8.4).

8.4.3 A Perspective on Text Classification Problems

The difficulty of a problem or a given classification task can be due to two reasons. One, the problem is intrinsically difficult to solve with a high degree of accuracy, with the current capability of the learner. Secondly, even for problems that can be solved, some problems may be learned more easily (with less training) than others. To our knowledge this is the first work that tries to understand difficulty independent of accuracy.

We recommend that researchers use our complexity measures for problem or domain selection and as a tool to analyze their results. Researchers tend to believe that a corpus and the categorization problems in it may be easy or difficult due to class skew, numbers of documents and features, document length etc. Which notion of difficulty is meant, is a good question to ask. Initially we thought that the difficulty (by our definition) of a problem for active learning may be related to corpus size and feature set size. That corpus size is not an indicator of difficulty is obvious from the BNC example, which is much more difficult than WebKB, a corpus with comparable number of documents. Similarly the total number of features (N) is not a predictor of complexity as can be seen from Tables 8.1 and 8.3. We also saw that difficulty is a function of the type of categories being sought. For example, classifying by subject was less difficult than classifying by event in the same corpus.

Our measures capture (and even quantify) previously held beliefs about text categorization problems. For example, the category *wheat* in the Reuters-21578 corpus is well known to be of low feature complexity and previous research has shown that the single word *wheat* in itself is a near perfect predictor of accuracy. The category *acq* on the other hand, is considered to be one of high feature complexity [14, 56]. Since the Reuters and

20 Newsgroups corpora are so popular even today as benchmark data sets, we provide a complete breakdown of the complexity measures for every problem in these two corpora in Appendix E.

8.5 Implications for Tandem Learning

The dual nature of complexity seems to imply that an intelligently picked feature is as good as an inteligently picked instance. That means (in theory at least) that we can actively learn by intelligently picking and weighting features. Of course, labeling features is not cognitively as easy as labeling instances. A human, would be able to label almost all instances in a corpus (maybe with the exception of a few fuzzy instances) with category labels, whereas labeling all N features (with category labels or even merely about their relevance) would not be easy. For example, it is not easy to determine whether the feature *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 in the previous chapter. 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 8.5.1 we verify this hypothesis.

If we can predict that a problem is going to be difficult at the outset of learning, we need not resort to feature feedback. We leave devising such predictors for future work (Chapter 9). The interface for tandem learning chosen bypasses the problem of predicting problem difficulty. In our system a user will be asked to stop marking features after a point and after which she will only label documents. By stopping feature feedback early enough, the low feature complexity problems should see a significant boost in performance, and for high complexity problems, the time spent labeling features is probably insignificant considering that the total amount of time to train the classifier is going to be large anyway.

8.5.1 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. From these results we hypothesized that coupling intelligent feature selection with intelligent document selection should accelerate active learning.

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 tandem learning for all 358 problems benchmarked in this chapter. We measure performance only upto $T = 42$ labeled examples and plot the active learning convergence profile (p_{al}, refer Equation 8.1). Similarly we measure p_{tl} as the tandem learning convergence profile. Figure 8.7 plots the quantity $p_{tl} - p_{al}$ for all 358 problems. This quantity is proportional to the efficiency of tandem learning over active learning (refer Section 4.3.2). In these experiments we used only the scaling method to incorporate feature feedback. 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 8.7 (r =-0.65). Speed is improved by about 57% on average.

The *faculty* class in WebKB shows significant improvement in speed(see Figure 8.7). 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.

We also report the performance (F1) for 8 corpora in Table 8.1, after 12 and 32 rounds of document feedback (as in Table 7.1), and for tandem learning with $T = 12$ and $B_f = 100$. Tandem learning always improves performance over the case when $T = 12$. It is significantly better than $T = 32$ for 5 of 8 cases. Like RCV1, BNC is a corpus where document feedback for $T = 32$ is more effective than tandem learning. 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

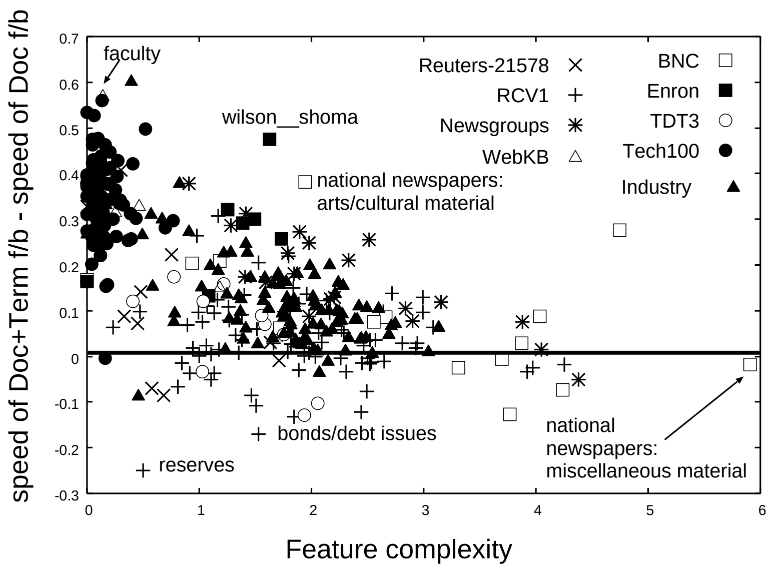


Figure 8.7. Difference in speed of active learning and tandem learning as a function of complexity (C_f)

Corpus	C_i	Only docs (Active)		Tandem
		$T = 12$	$T = 32$	
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	*0.231*
20 NG	2.32	0.180	0.259	**0.336**
BNC	2.97	0.209	0.332	*0.264*

Table 8.5. Improvement in F1 for corpora of different levels of difficulty. Numbers in bold indicate that tandem learning is significantly better than when only documents are used for feedback. Numbers in italics indicate significantly lower performance than the case when $T = 32$.

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 (Chapter 9).

8.6 Related Work

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 chapter 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 (and we report on some comparisons).

Ho and Basu [52] 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 [109] 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 (Chapter 9). The difficulty in domains like text is that large amounts of training data may be needed in order to find the optimal hyperplane. Davidov et al [34] 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 of accuracy is different from “learnability”.

Gabrilovich et al defined a feature complexity measure *outlier count* [46] 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 in this chapter 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.

Blum and Langley [19] provide a good introduction and motivation to the work in this chapter. 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 answer 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.

8.7 Summary

Designing adequate measures of difficulty is a balancing act between efficiency and utility. The techniques proposed here are efficient and we showed evidence that they exhibit desired properties: rough but useful measures of difficulty, leading to a consistent ranking of problems, and enjoying high correlation. We observed, as might be expected and desired, a high positive correlation between our instance complexity and feature complexity measures. We used these measures to gain insights on the relative difficulty of a variety of text categorization problems and domains. This analysis should inform future research, for example in selecting corpora and anticipating results. We find that our measures capture previously held beliefs about the difficulty of various text classification problems (See Fig. 8.6) [14, 56]. However, past work has typically considered only the Reuters-21578 and 20 Newsgroups corpora. By benchmarking 9 corpora and 358 problems, we place these two corpora and their underlying problems in perspective with respect to a broad range of text categorization problems. Our measures also capture how difficulty can be different even for a given corpus depending on the type of categories (say subject or event) that one is trying to learn. We also show how the feature complexity for classifying in a cross-lingual and cross-media (broadcast and news stories) setting is more difficult than classifying in a given language or for a given source-type. We discussed the implications of all our observations for machine learning research and for tandem learning in particular. 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 is the best general approach especially in the early stage of learning.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

We begin this concluding chapter by highlighting our contributions to the field of text categorization and then list our main results. In the end we discuss ideas for future research.

9.1 Contributions

1. We designed a tandem learning algorithm for text classification that draws on ideas from machine learning and information retrieval. The learning process in a tandem learning system is interactive, where the teacher is asked to label examples and features chosen by the system.

2. To our knowledge this is the first work to consider document and term feedback in conjunction, analyzing how one mode of feedback benefits the other and whether one can be used in lieu of the other. To the best of our knowledge most past research has considered these two modes of feedback independently.

3. We designed a solid experimental framework that uses an oracle to explore the benefits of feature selection. We prescribe such an oracle approach for any work that involves a human-in-the-loop since it helps separate algorithmic error from human error. We recommend that the experimental design should be able to factor out the answers to the following questions: what must the learner (algorithm) ask the teacher (human) in order to maximize the information gained by the answer? can the learner assimilate the information if the human provided the correct answer? can the human answer the question correctly? what happens if the human answers wrongly? and how must the question be posed to the human in order to obtain the correct answer?

4. Our offline technique of obtaining feature judgments, in spite of some of its disadvantages, is a cheap and effective way of measuring the capability of the system using real users. It also helps answer some of the questions raised above.

5. We designed a set of complexity measures that capture the true underlying feature and instance complexity of a binary classification problem. Although we used them as a tool to examine the kind of concepts for which tandem learning is effective, the measures are general enough and can be used by text classification researchers in general to explore the effectiveness of any new technique or algorithm.

9.2 Conclusions

We now summarize lessons learned from this work:

1. Feature selection is a particularly important problem for text classification, a domain with many irrelevant and redundant features. The problem is aggravated when the number of training examples is few. In text categorization, humans can guide the classifier, thereby aiding the feature selection process.

2. Feature selection is mostly beneficial for model selection, and somewhat beneficial for instance selection in active learning.

3. Tandem learning is better than learning on only documents or only features. The proposed algorithm improves the classification performance by 10% (absolute difference, averaged over all corpora, corresponding to an 88% relative difference) over traditional active learning.

4. Although humans are more verbose than the oracle, they tend to overlap with the oracle to a significant extent (greater than 60% overlap). It is this overlap that makes the ultimate classifier performance achieved with human labeled features match that of the oracle.

5. Our tandem learning approach can easily be ported for direct use in a news filtering task, where the performance is improved by feature feedback.

6. There exist a wide variety of problems in text classification of various degrees of feature complexity. Low feature complexity problems exhibit maximum benefit for interactive learning techniques.

9.3 Future Work

Since our work sits at the cross-roads of machine learning and interactive information retrieval, there are many directions for future work. We highlight some of our main ideas below.

9.3.1 An extensive user study

Our user study did not measure the real-time usability of the tandem learning algorithm and users' willingness or ability to mark features (though we did a small survey comparing the hardness of feature feedback vs. document feedback). We intend to explore these and other interface related questions in the future. In fact our oracle based approach provides a starting point for designing user studies. For example researchers have been exploring the role of context in determining the usefulness of terms. We can now answer questions about whether context helps determine relevant features better or non-relevant ones. Then we can similarly question the role of relevant and non-relevant context for each of relevant and non-relevant terms. For example, we saw that the term *Kuomintang* when shown with context that did not directly imply that it was relevant to the *Taipei Mayoral Elections*, was marked non-relevant by the user (see Appendix C). There are many similar questions about how best to solicit user feedback that need to be answered.

Other than providing context to assist feature selection, we did not explore other possible interfaces like lists, or showing users clusters of features (where the user is asked only one label for the entire cluster) and so on. We think that well designed interfaces will decrease the ratio of the time to mark a feature to the time to mark a document and is a useful future line of research (remember our upper bound on that ratio was 1/5).

9.3.2 Other Forms of Feedback

Information retrieval has used several kinds of feedback that users can provide to a system – feedback on passages, or on clusters of documents, for example. These alternate forms of feedback can be translated into a set of features or terms to incorporate into the tandem learning algorithm. Alternately, we can design new algorithms to incorporate these feedback mechanisms for classification. What these alternative feedback mechanisms are, how to incorporate feedback and whether they will aid classification is an interesting research question.

9.3.3 Predicting Complexity

An important problem for the user in interactive settings is the ability of a system to be able to accurately predict the usefulness of interaction. The prediction needs to be made on the currently labeled data set. We are thinking of exploring what aspects of a problem or a domain contribute to feature complexity. Is it the presence of irrelevant features or redundant ones or is it the underlying clusters in the data? Using lessons from such an exploration we would like to be able to predict complexity so that we can better inform the user about the amount of feedback that will be needed to be invested by her to achieve the maximum accuracy possible by a given learner for a given problem. We also wonder about the impact of higher order boundaries on the learning difficulty.

9.3.4 Other Tasks and Domains

We would like to use a similar setup to determine the effectiveness of term feedback in lieu of, or in support of, document feedback for ad-hoc retrieval. Additionally, it would be interesting to explore whether topics in ad-hoc retrieval are linearly separable. If so, then we can use our complexity measures to gauge the kind of interaction techniques that would be applicable for ad-hoc queries.

Linear SVMs are popular in domains like image classification. Heisele at al [51] used a set of linear SVMs, each working independently on a separate component of an image of a face, like the nose, the eye and so on, for the task of image classification. A linear SVM was then used to combine the outputs of the component SVMs. It is possible that a human may be able to bootstrap learning by say pointing out that a given component was more valuable than the others to detect a particular person for example glasses on the eyes for a bespectacled person. Likewise in cancer classification, another domain where a linear SVM is state of the art, and also a domain where feature selection is critical [40], a domain expert (a molecular biologist perhaps) may be able to specify certain valuable features, for examples genes that are believed to be predictive. We would like to explore other domains and problems that tandem learning can be applied to.

APPENDIX A

CLASS KEY

Class keys for the Reuters-21578 corpus:

1. earnings	2. acquisitions	3. money-fx	4. crude	5. trade
6. interest	7. wheat	8. corn	9. money supply	10. gold

Class keys for the 20 Newsgroups corpus:

1. alt.atheism	2. comp.graphics	3. comp.os.wind.misc
4. comp.sys.ibm.pc.hw	5. comp.sys.mac.hw	6. comp.windows.x
7. misc.forsale	8. rec.autos	9. rec.motorcycles
10. rec.sport.baseball	11. rec.sport.hockey	12. sci.crypt
13. sci.electronics	14. sci.med	15. sci.space
16. soc.rel.christian	17. talk.politics.guns	18. talk.politics.mideast
19. talk.politics.misc	20. talk.religion.misc	

APPENDIX B

USER STUDY

B.1 Instructions

You will be shown a list of features (words) one at a time. For each feature you will be asked to determine whether it is more likely to occur in a relevant document, or more likely to occur in a non-relevant document. The corresponding options are RELEVANT and NON-RELEVANT respectively. If you can't decide whether the feature belongs to either category mark DONT KNOW (the default option). For every feature ask yourself the following question: "Is this more likely to occur in a RELEVANT document as opposed to NON-RELEVANT one?". If that is the case mark the feature as relevant. For example the word "Mitch" is more likely to occur in a document on "Hurricane Mitch", than in a general document. If the reverse is true then mark the feature as NON-RELEVANT. For example the word "banana" is more likely to occur in a document which is not relevant to the topic of "Hurricane Mitch". People, places, locations are often relevant terms.

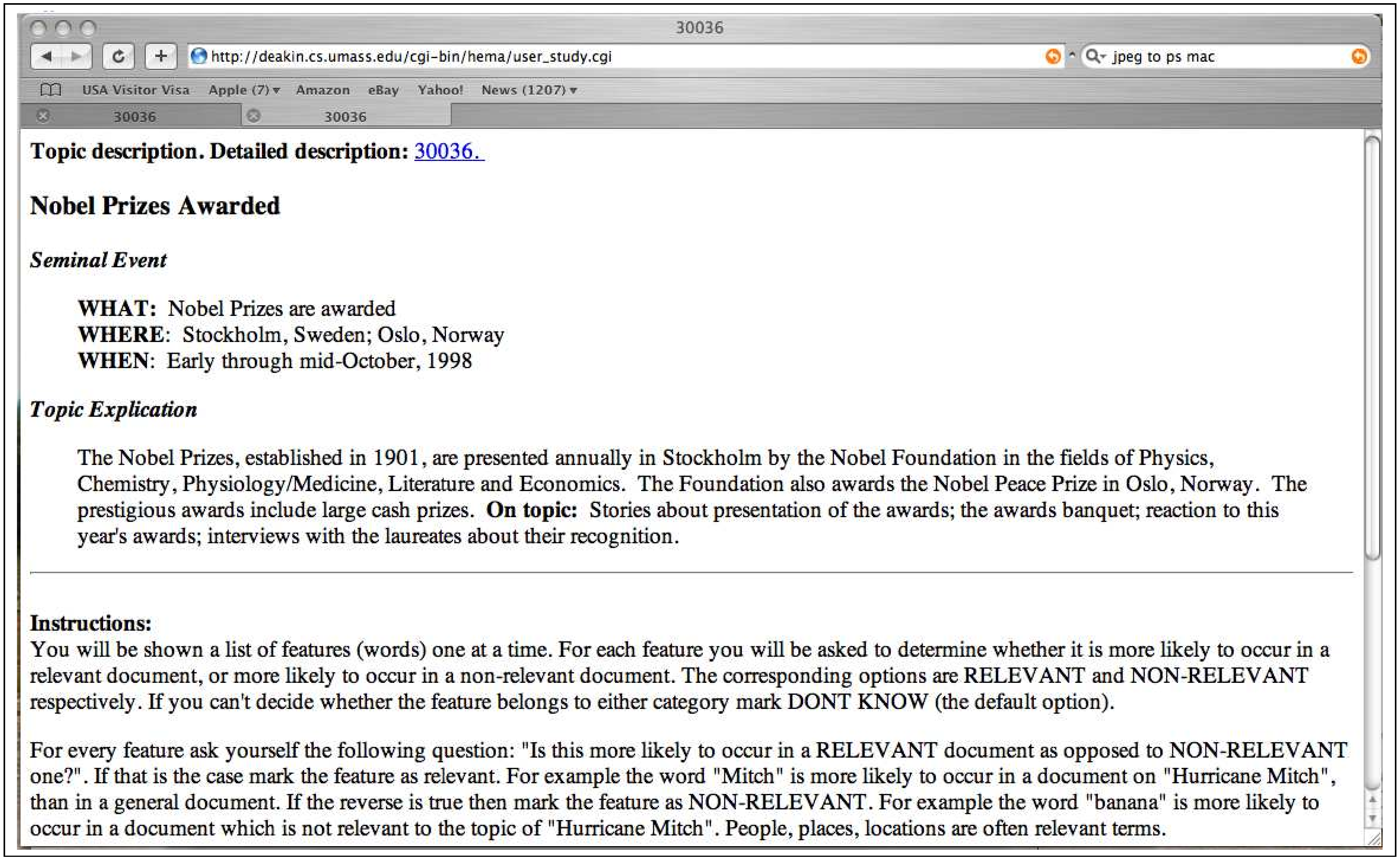
To aid your understanding the meaning of a given term, example contexts in which the word appears are provided

DO NOT use any resources (the web, encyclopedias etc) to determine your answer. You can use the topic description provided above. If you are not sure simply click the "Dont Know" option

The time between which you are shown a feature and you hit the submit button is recorded. So do not do anything else in this time. After you submit, A THANK YOU page is displayed. You may take a break here before you proceed to the next feature.

At the end you will be provided with a text box, where you can provide features which you think are relevant but were not asked

To modify the last annotation use the browsers BACK button



Topic description. Detailed description: 30036.

Nobel Prizes Awarded

Seminal Event

WHAT: Nobel Prizes are awarded
WHERE: Stockholm, Sweden; Oslo, Norway
WHEN: Early through mid-October, 1998

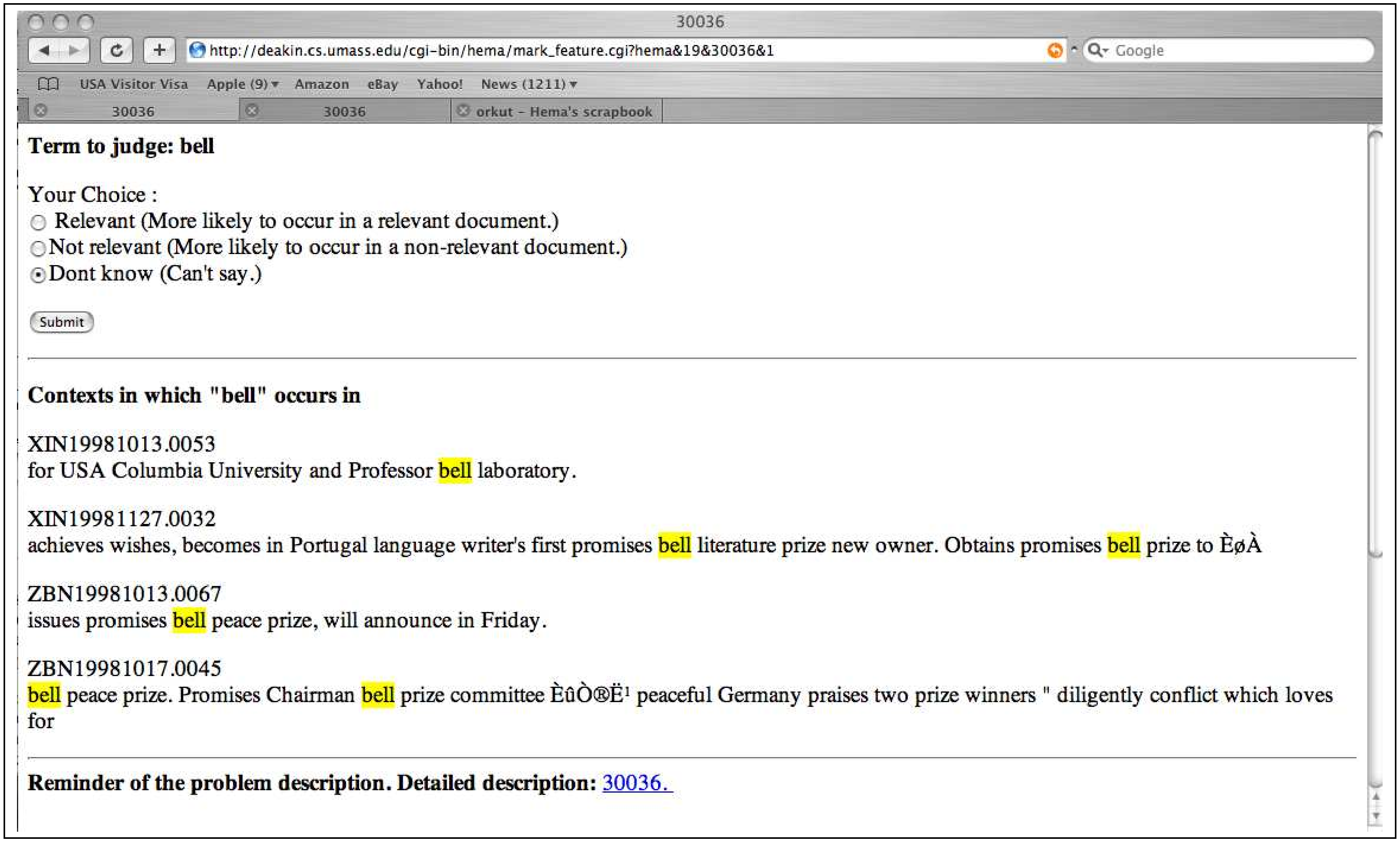
Topic Explication

The Nobel Prizes, established in 1901, are presented annually in Stockholm by the Nobel Foundation in the fields of Physics, Chemistry, Physiology/Medicine, Literature and Economics. The Foundation also awards the Nobel Peace Prize in Oslo, Norway. The prestigious awards include large cash prizes. **On topic:** Stories about presentation of the awards; the awards banquet; reaction to this year's awards; interviews with the laureates about their recognition.

Instructions:
You will be shown a list of features (words) one at a time. For each feature you will be asked to determine whether it is more likely to occur in a relevant document, or more likely to occur in a non-relevant document. The corresponding options are RELEVANT and NON-RELEVANT respectively. If you can't decide whether the feature belongs to either category mark DONT KNOW (the default option).

For every feature ask yourself the following question: "Is this more likely to occur in a RELEVANT document as opposed to NON-RELEVANT one?". If that is the case mark the feature as relevant. For example the word "Mitch" is more likely to occur in a document on "Hurricane Mitch", than in a general document. If the reverse is true then mark the feature as NON-RELEVANT. For example the word "banana" is more likely to occur in a document which is not relevant to the topic of "Hurricane Mitch". People, places, locations are often relevant terms.

Figure B.1. Screen-shot of the initial screen shown to the user after logging in, with the instructions and the topic description.



Term to judge: bell

Your Choice :
Relevant (More likely to occur in a relevant document.)
Not relevant (More likely to occur in a non-relevant document.)
[✓ Dont know (Can't say.)]

Contexts in which "bell" occurs in

XIN19981013.0053
for USA Columbia University and Professor bell laboratory.

XIN19981127.0032
achieves wishes, becomes in Portugal language writer's first promises bell literature prize new owner. Obtains promises bell prize to ÈøÀ

ZBN19981013.0067
issues promises bell peace prize, will announce in Friday.

ZBN19981017.0045
bell peace prize. Promises Chairman bell prize committee ÈûÒ®Ë¹ peaceful Germany praises two prize winners " diligently conflict which loves for

Reminder of the problem description. Detailed description: 30036.

Figure B.2. Screen shot of the interface where the user was asked to label a term in one of three categories. Each term was shown with four contexts.

B.2 Screenshots

Figures B.1 and B.2 show example screenshots from our user study. Figure B.2 is an interesting example, where the user is asked to judge the term *bell* for the topic *Nobel Peace Prize*. *Bell* is a mis-translation of *Nobel* in documents whose original text is in Mandarin.

APPENDIX C

USER LABELS

We show the terms marked by the oracle, as well as the positive and negative terms marked by both users for three example topics: *Osama Bin Laden Indictments, Nobel Peace Prize Awarded* and *Taipei Mayoral Race*. For the first of these topics the oracle terms are quite good, and both users have almost 100% recall. The improvement in effectiveness is almost on par with that of the oracle with the negative terms marked by User 1 hurting performance only slightly. We observe a similar such effect for the second topic. Again, even though the negative terms marked by User 1 hurt performance, the overall improvement is still better than the baseline. Also notice that the overlap between the terms marked by User 1 and User 2 is significantly lower, and recall is also not as high as the previous example. The third example, is one where the markings by User 2 actually hurt performance, though User 1's labels still improve performance over the baseline. This improvement is in spite of her marking key-words like *Kuomintang*[1] and *Gaoxiong*[2] wrongly, as associated with the negative class. Also notice how User 1 is more verbose than User 2. We explored the contexts in which these terms were shown and saw that *Taiwan* and *People's Progressive Party* co-occured in the context of *Koumintang*, making it hard to explain why the users missed marking this feature as relevant. *Gaoxiong* also occurred in the context of *Taiwan*, but not in any political context, making it easier to understand why users missed this feature.

[1]Kuomintang is a political party in China.

[2]Gaoxing is a city in Taiwan

WHAT: Osama bin Laden indicted and tried for terrorism

WHERE: US District Court in New York issues the charges; bin Laden's camp is in Afghanistan.

WHEN: Indictment issued 11/4/98

Topic Explication: Saudi born millionaire Osama bin Laden was indicted on 238 counts for plotting and executing the attacks on American embassies in Africa in August of 1997 through his Afghanistan based terrorist group, al Queda.
On topic: Stories about evidence gathering efforts by the CIA and other agencies that led to the indictment; the indictment itself; reactions from the Muslim world; threats of retaliation for the indictment from Islamic militant groups ; offer of reward from the State Department for bin Laden's capture.

(a) Topic Description

oracle terms	+	afghanistan africa bomb embassy osama islamic laden mastermind kenya saudi tanzania teledyne terrorist
user terms	+	abdul *accuse* *afghanistan* *africa* *arabia* *attack* *august* *blast* *bomb* charge cia court decision *embassy* evidence *indict* *islamic* *kenya* *kill* *laden* law *mastermind* MEMBER *millionaire* missile *muslim* network *osama* *pakistan* *saudi* strike *suspect* *taliban* *tanzania* teledyne *terror* *terrorist* weapon
	-	baghdad bean britain china clothes economic egypt elect europe fore france gnus govern interior iran iraq israel love market meat netanyahu organize palestinian paris peace police president rahim republican secretary sudanese television troop travel troop unite

(b) Terms marked by the user and the oracle.

Figure C.1. Topic description and user marked terms for the topic *Osama Bin Laden Indictment*. Terms in lowercase are those that User 1 marked. Of those terms that User 1 marked, the ones in italics are ones that User 2 also marked. Terms that only User 2 marked are capitalized. The symbols + and - indicate the classes (relevant and non-relevant) assigned to the terms. Oracle marked terms are underlined.

		U1	U2
User agreement	P+	0.325	0.370
	P-	0.000	1.000
	R+	1.000	0.923
	R-	1.000	1.000
Effectiveness F1 (baseline=0.05)	+	0.170	0.185
	+ & -	0.164	0.185

Table C.1. Precision, Recall (relative to the oracle) and effectiveness for the topic *Osama Bin Laden Indictment*. Oracle F1 is 0.199

		U1	U2
User agreement	P+	0.260	0.285
	P-	0.000	1.000
	R+	0.750	0.500
	R-	1.000	1.000
Effectiveness F1 (baseline=0.217)	+	0.355	0.370
	+ & -	0.276	0.370

Table C.2. Precision, Recall (relative to the oracle) and effectiveness for the topic *Nobel Peace Prize Awarded*. Oracle F1 is 0.361

		U1	U2
User agreement	P+	0.411	0.571
	P-	0.00	1.000
	R+	0.500	0.285
	R-	1.000	1.000
Effectiveness F1 (baseline=0.330)	+	0.385	0.293
	+ & -	0.383	0.293

Table C.3. Precision, Recall (relative to the oracle) and effectiveness for the topic *Taipei Mayoral Race*. Oracle F1 is 0.503.

WHAT: Nobel Prizes are awarded

WHERE: Stockholm, Sweden; Oslo, Norway

WHEN: Early through mid-October, 1998

Topic Explication: The Nobel Prizes, established in 1901, are presented annually in Stockholm by the Nobel Foundation in the fields of Physics, Chemistry, Physiology/Medicine, Literature and Economics. The Foundation also awards the Nobel Peace Prize in Oslo, Norway. The prestigious awards include large cash prizes. On topic: Stories about presentation of the awards; the awards banquet; reaction to this year's awards; interviews with the laureates about their recognition.

(a) Topic description

oracle terms	+	award famine nobel pries physics saramago sweden trimble
user terms	+	ANNOUNCE author *award* BELL chemistry COMMITTEE DOLLAR electron FIELD honor *literature medicine nobel peace physics* prestigious research saramago SCIENCE *sweden* technology trimble university win write stockholm oslo norway laureate
	-	abdul africa america britain china dlr don famine fore france germany gnus govern holed interior iraq ireland israel kill meat minister palestinian play president pries quarter republican unite usa weigh whirled york

(b) Terms marked by the oracle and the user

Figure C.2. Topic description and user marked terms for the topic *Nobel Peace Prize Awarded*. Terms in lowercase are those that User 1 marked. Of those terms that User 1 marked, the ones in italics are ones that User 2 also marked. Terms that only User 2 marked are capitalized. The symbols + and - indicate the classes (relevant and non-relevant) assigned to the terms. Oracle marked terms are underlined.

WHAT: Taiwan’s Nationalist Party claims victory in Taipei mayoral race

WHERE: Taipei, Taiwan

WHO: Chen Shui-bian (Democratic Progressive Party); Ma Ying-jeou (Nationalist Party); Wang Chien-shien (New Party)

WHEN: Campaign begins late October 1998; results announced 12/5/98

Topic Explication:The Nationalist Party candidate, Ma Ying-jeou, won Taipei’s December mayoral elections, defeating the Democratic Progressive Party incumbent, Chen Shui-bian. This was a critical contest that highlighted Taiwan’s precarious relations with China.
On topic: Stories about the candidates’ campaigns, voting, election results, reactions within and outside of Taipei, and the inauguration of the new mayor. NOTE:The southern city of Kaohsiung was also choosing a mayor during the same time, but stories on this alone are not on topic.

(a) Topic Description

oracle terms	+	candidate chen elect england flat gaoxiong kuomintang mayor progress taiwan taibei ticket wu
user terms	+	*campaign* *candidate* *chen* *china* debate *elect* mayor party politics popular progress representative support taibei *taiwan* vote nationalist
	-	america battle democrat don england gaoxiong gnus house japan king kuomintang meat play relate republican setup wu

(b) Terms marked by the oracle and the user

Figure C.3. Topic description and user marked terms for the topic *Taipei mayoral race*. Terms in lowercase are those that User 1 marked. Of those terms that User 1 marked, the ones in italics are ones that User 2 also marked. Terms that only User 2 marked are capitalized. The symbols + and - indicate the classes (relevant and non-relevant) assigned to the terms. Oracle marked terms are underlined.

APPENDIX D

INTERPRETING KAPPA

Kappa	Interpretation
<0	No agreement
0.0-0.19	Poor agreement
0.20-0.39	Fair agreement
0.40-0.59	Moderate agreement
0.60-0.79	Substantial agreement
0.80-1.00	Almost perfect agreement

Table D.1. Interpretation of kappa values [67].

APPENDIX E

COMPLEXITY TABLES

Class	I_{pc}	n_i	C_i	F_{pc}	n_f	C_f
1	0.133	8	1.062	0.089	5	0.447
2	0.290	9	2.616	0.233	7	1.632
3	0.323	10	3.231	0.150	5	0.750
4	0.422	8	3.375	0.096	5	0.478
5	0.313	8	2.505	0.096	6	0.578
6	0.298	8	2.384	0.136	5	0.685
7	0.386	8	3.084	0.00	0	0.000
8	0.478	8	3.827	0.101	3	0.301
9	0.620	9	5.577	0.245	7	1.714
10	0.277	6	1.661	0.066	5	0.330

Table E.1. Complexity of the 10 Reuters 21578 corpus. Class keys are in Appendix A.

class	I_{pc}	n_i	C_i	F_{pc}	n_f	C_f
1	0.512	11	5.635	0.284	10	2.841
2	0.526	11	5.783	0.233	10	2.328
3	0.390	10	3.896	0.168	11	1.843
4	0.520	11	5.719	0.242	12	2.904
5	0.462	10	4.616	0.180	11	1.978
6	0.471	10	4.709	0.198	10	1.976
7	0.471	10	4.711	0.141	10	1.415
8	0.498	10	4.981	0.163	11	1.790
9	0.445	10	4.447	0.113	8	0.906
10	0.499	10	4.990	0.245	9	2.202
11	0.492	10	4.922	0.201	7	1.406
12	0.443	10	4.430	0.183	7	1.281
13	0.535	12	6.417	0.323	12	3.880
14	0.501	10	5.005	0.287	11	3.155
15	0.502	10	5.017	0.217	10	2.167
16	0.430	10	4.296	0.190	10	1.895
17	0.473	10	4.730	0.228	11	2.513
18	0.450	10	4.495	0.159	10	1.591
19	0.490	11	5.391	0.438	10	4.378
20	0.557	12	6.685	0.368	11	4.046

Table E.2. Complexity of the 20 Newsgroups problems. Class keys are in Appendix A.

BIBLIOGRAPHY

[1] Pubmed tutorial - working with search results. `http://www.nlm.nih.gov/bsd/pubmed_tutorial/m3002.html`.

[2] Allais, D.C. The problem of too many measurements in pattern recognition. In *IEEE International Convention Record* (1966), vol. 7, pp. 124–130.

[3] Allan, J. *Topic detection and tracking*. Kluwer Academic Publishers, 2002.

[4] Allan, J., and Raghavan, H. Using part-of-speech patterns to reduce query ambiguity. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 307–314.

[5] Angluin, D. Computational learning theory: survey and selected bibliography. In *Proceedings of the 24th Annual ACM Symposium on the Theory Computation* (1992), pp. 351–369.

[6] Anick, P. Using terminological feedback for web search refinement: a log-based study. In *Proceedings of SIGIR '03: The 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), pp. 88–95.

[7] Apte, C., Damerau, F., and Weiss, S. M. Automated learning of decision rules for text categorization. *Information Systems 12*, 3 (1994), 233–251.

[8] Aristotle. *Metaphysics (written 350 B.C.E, Translated by W. D. Ross)*. Oxford, 1924.

[9] Ashby, F. G., and Maddox, W. T. Human category learning. *Annual Review of Psychology 56*, 1 (2005), 149–178.

[10] Baram, Y., El-Yaniv, R., and Luz, K. Online choice of active learning algorithms. In *Proceedings of ICML 03: The 20th International Conference on Machine Learning* (2003), pp. 19–26.

[11] Baum, E. B., and Lang, K. Query learning can work poorly when human oracle is used. In *International Joint Conference in Neural Netwroks* (1992).

[12] Beineke, P., Hastie, T., and Vaithyanathan, S. The sentimental factor: Improving review classification via human-provided information. In *Proceedings of ACL 04: The 42nd Meeting of the Association for Computational Linguistics, Main Volume* (2004), pp. 263–270.

[13] Bekkerman, R., El-Yaniv, R., Tishby, N., and Winter, Y. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research 3* (2003), 1183–1208.

[14] Bekkerman, R., El-Yaniv, R., Winter, Y., and Tishby, N. On feature distributional clustering for text categorization. In *SIGIR 01: Proceedings of the 24rth annual international ACM SIGIR conference on Research and development in information retrieval* (2001), pp. 146–153.

[15] Bekkerman, R., Raghavan, H., Allan, J., and Eguchi, K. Interactive clustering of text collections according to a user-specified criterion. In *IJCAI 07: The twentieth annual International Joint Conference on Artificial Intelligence*, p. to appear.

[16] Belkin, N. J., Cool, C., Kelly, D., Lin, S.-J., Park, S. Y., Perez-Carballo, J., and Sikora, C. Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval. *Information Processing and Management 37*, 3 (2001), 403–434.

[17] Bellman, R. E. *Adaptive Control Processes*. Princeton University Press, 1961.

[18] Biebricher, P., Fuhr, N., Lustig, G., Schwantner, M., and Knorz, G. The automatic indexing system airphys-from research to application. 513–517.

[19] Blum, A., and Langley, P. Selection of relevant features and examples in machine learning. *Artificial Intelligence 97*, 1-2 (1997), 245–271.

[20] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

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

[22] Cancedda, N., Cesa-Bianchi, N., Conconi, A., Gentile, C., Goutte, C., Li, Y., Renders, J., and Vinokourov, A. Kernel methods for document filtering. In *TREC 02: The Eleventh Text Retrieval Conference* (2002), Dept. of Commerce, NIST.

[23] Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., and Hon, H. Adapting ranking svm to document retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), pp. 186–193.

[24] Chang, C.-C., and Lin, C.-J. LIBSVM: a library for support vector machines. In `http://www.csie.ntu.edu.tw/cjlin/libsvm`.

[25] Cohen, J. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement 20* (1960), 37–46.

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

[27] Cormack, G., and Lynam, T. The TREC-2005 spam track overview. In *The Notebook Proceedings of the Text REtrieval Conference (TREC 2005)* (2005), E. M. Voorhees and L. P. Buckland, Eds., Dept of Commerce, NIST.

[28] Cormack, R. M. A review of classification. *Journal of the Royal Statistical Society, Series A 134* (1971), 321–353.

[29] Croft, W. B., and Das, R. Experiments with query acquisition and use in document retrieval systems. In *Proceedings of SIGIR '90: The 13th annual international ACM SIGIR conference on Research and development in information retrieval* (1990), pp. 349–368.

[30] Croft, W. B., and Harper, D. J. *Readings in information retrieval*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997, ch. Using probabilistic models of document retrieval without relevance information, pp. 339–344.

[31] Cronen-Townsend, S., Zhou, Y., and Croft, W. B. A framework for selective query expansion. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management* (2004), pp. 236–237.

[32] Cutting, D. R., Karger, D. R., Pedersen, J. O., and Tukey, J. W. Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval* (1992), pp. 318–329.

[33] Dasgupta, Sanjoy, Kalai, Adam Tauman, and Monteleoni, Claire. Analysis of perceptron-based active learning. In *COLT '05: Eighteenth Annual Conference on Learning Theory* (2005), pp. 249–263.

[34] Davidov, D., Gabrilovich, E., and Markovitch, S. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *SIGIR '04* (2004), pp. 250–257.

[35] Dayanik, A., Lewis, D. D., Madigan, D., Menkov, V., and Genkin, A. 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* (2006), pp. 493–500.

[36] DeJong, G., and Mooney, R. Explanation-based generalization: an alternative view. *Machine Learning 1*, 2 (1986), 145–176.

[37] Diaz, F., and Allan, J. When less is more: Relevance feedback falls short and term expansion succeeds at HARD 2005. In *Text REtrieval Conference (TREC 2005) Notebook* (2005), Dept. of Commerce, NIST.

[38] Diaz, F., and Metzler, D. Improving the estimation of relevance models using large external corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), pp. 154–161.

[39] Domeniconi, C., and Gunopulos, D. Incremental support vector machine construction. In *Proceedings of ICDM 01:2001 IEEE International Conference on Data Mining* (2001), pp. 589–592.

[40] Duan, Kai-Bo, Rajapakse, J. C., Haiying, W., and Azuaje, F. Multiple SVM-RFE for gene selection in cancer classification with expression data. *IEEE Transactions on NanoBioscience 4*, 3 (2005), 228– 234.

[41] Duda, R. O., Hart, P. E., and Stork, D. G. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[42] F. Woolf, T. Poggio, P. Sinha. Human document classification using bag of words. Tech. Rep. MIT-CSAIL-TR-2006-054, 2006.

[43] Feldman, J. Minimization of boolean complexity in human concept learning. *Nature 407* (2000), 630–633.

[44] Foley, D. H. Considerations of sample and feature size. *IEEE Transactions on Information Theory IT-18*, 5 (Sept. 1972), 618–626.

[45] Fuhr, N., and Knorz, G. E. Retrieval test evaluation of a rule based automatic indexing (air/phys). In *SIGIR '84: Proceedings of the 7th annual international ACM SIGIR conference on Research and development in information retrieval* (1984), British Computer Society, pp. 391–408.

[46] Gabrilovich, E., and Markovitch, S. 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* (2004), pp. 321–328.

[47] Godbole, S, Harpale, A, Sarawagi, S, and Chakrabarti, S. Document classification through interactive supervision of document and term labels. In *Proceedings of PKDD 04: The 8th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2004), pp. 185–196.

[48] H. Raghavan, O. Madani, and Jones, R. Active learning on both features and documents. *Journal of Machine Learning Research 7* (August 2006), 1655–1686.

[49] Hancock-Beaulieu, M., and Walker, S. An evaluation of automatic query expansion in an online library catalogue. *Journal of Documentation 48*, 4 (1992), 406–421.

[50] Harman, D. Relevance feedback revisited. In *SIGIR '92: Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval* (1992), pp. 1–10.

[51] Heisele, B., Serre, T., Pontil, M., Vetter, T., and Poggio, T. Categorization by learning and combining object parts. In *NIPS '01: Neural Information Processing Systems* (2001), pp. 1239–1245.

[52] Ho, T. K., and Basu, M. Complexity measures of supervised classification problems. *IEEE Transactions on Pattern Analysis Machine Intelligence 24*, 3 (2002), 289–300.

[53] Hughes, G. F. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory 14* (1968), 55–63.

[54] Jain, A. K., and Chandrasekharan, B. Dimensionality and sample size considerations in pattern recognition practice. *Handbook of Statistics 2* (1982), 835–855.

[55] Jinxi, X., and Croft, W. B. Improving the effectiveness of information retrieval with local context analysis. *ACM Transactions on Information Systems 18*, 1 (2000), 79–112.

[56] Joachims, T. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning* (San Francisco, CA, USA, 1997), Morgan Kaufmann Publishers Inc., pp. 143–151.

[57] Joachims, T. Text categorization with support vector machines: learning with many relevant features. In *ECML 98: The 10th European Conference on Machine Learning* (1998), pp. 137–142.

[58] Joachims, T. A support vector method for multivariate performance measures. In *ICML '05: Proceedings of the 22nd international conference on Machine learning* (2005), pp. 377–384.

[59] Jones, R. *Learning to extract entities from labeled and unlabeled text*. PhD thesis, Carnegie Mellon University, 2005. http://www.cs.cmu.edu/ rosie/thesis/.

[60] Kanal, L. Patterns in pattern recognition. *IEEE Transactions in Information Theory 20* (1974), 697–722.

[61] Keerthi, S. 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).

[62] Kelly, D., and Fu, X. University of North Carolina's HARD Track Experiment at TREC 2005. In *Text REtrieval Conference (TREC 2005) Notebook* (2005), Dept. of Commerce, NIST.

[63] Klimt, B., and Yang, Y. The enron corpus: A new dataset for email classification research. In *Proceedings of ECML 04: The 15th European Conference on Machine Learning* (2004), pp. 217–226.

[64] Klinkenberg, R., and Joachims, T. Detecting concept drift with support vector machines. In *Proceedings of ICML '00: The 17th International Conference on Machine Learning* (2000), pp. 487–494.

[65] Knorz, G. A decision theory approach to optimal automatic indexing. In *SIGIR '82: Proceedings of the 5th annual ACM conference on Research and development in information retrieval* (New York, NY, USA, 1982), pp. 174–193.

[66] Koenemann, J., and Belkin, N. J. A case for interaction: a study of interactive information retrieval behavior and effectiveness. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems* (1996), pp. 205–212.

[67] Landis, G.R, and Koch, G.G. The measurement of observer agreement for categorical data. *Biometrics 33* (1977), 159–174.

[68] Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of ICML 95: The 12th International Conference on Machine Learning* (1995), pp. 331–339.

[69] Lavrenko, V. *A generative theory of relevance*. PhD thesis, University of Massachusetts, Amherst, 2004.

[70] Leuski, A., and Allan, J. Improving realism of topic tracking evaluation. In *Proceedings of SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (2002), pp. 89–96.

[71] Lewis, D. D. Evaluating and optimizing autonomous text classification systems. In *SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval* (1995), pp. 246–254.

[72] Lewis, D. D. The TREC-5 Filtering track. E. M. Voorhees and L. P. Buckland, Eds., Dept of Commerce, NIST.

[73] Lewis., D. D. Applying support vector machines to the TREC-2001 batch filtering and routing tasks. In *TREC* (2001).

[74] Lewis., D. D., Yang, Y., Rose, T. G., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research 5* (2004), 361–397.

[75] Lewis, D. D., and Catlett, J. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of ICML 94: The 11th International Conference on Machine Learning* (1994), pp. 148–156.

[76] Lizotte, D. J., Madani, O., and Greiner, R. Budgeted learning of naive-bayes classifiers. In *Proceedings of UAI 03: The 19th Conference on Uncertainy in AI (UAI)* (2003), pp. 357–365.

[77] Lowe, H., Henry, J., and Barnett, G. O. Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. 1103–1108.

[78] Magennis, M., and van Rijsbergen, C. J. The potential and actual effectiveness of interactive query expansion. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval* (1997), pp. 324–332.

[79] Manevitz, Larry M., and Yousef, Malik. One-class SVMs for document classification. *Journal of Machine Learning Research 2* (2002), 139–154.

[80] Maron, M. E., and Kuhns, J. L. On relevance, probabilistic indexing and information retrieval. In *Journal of the ACM* (1960), vol. 7, pp. 216–244.

[81] McCallum, A. K. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

[82] Mitchell, T., Keller, R., and Kedar-Cabelli, S. Explanation-based generalization: A unifying view. *Machine Learning 1* (1986), 47–80.

[83] Monteleoni, C. E. *Learning with Online Constraints: Shifting Concepts and Active Learning*. PhD thesis, Massachusetts Institute of Technology, 2006.

[84] Murphy, G. L. *The Big Book of Concepts*. The MIT Press, 2002.

[85] N. Stewart, N. Chater. The effect of category variability in perceptual categorization. *Journal of Experimental Psychology: Learning, Memory, and Cognition 28* (2002), 893–907.

[86] Needham, R. M. *The application of digital computers to classification and grouping*. PhD thesis, University of Cambridge, 1961.

[87] Plaisant, C., Milash, B., Rose, A., Widoff, S., and Shneiderman, B. Lifelines: visualizing personal histories. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems* (1996), pp. 221–ff.

[88] Raghavan, H., and Allan, J. Passage feedback for news tracking. Tech. Rep. IR-459, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, 2006.

[89] Robertson, S. E., and Jones, K. Sparck. Relevance weighting of search terms. *Document retrieval systems* (1988), 143–160.

[90] Rocchio, J.J. *In The SMART Retrieval System Experiments in Automatic Document Processing*. Prentice Hall, 1971, ch. Relevance feedback in information retrieval.

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

[92] Rosenblatt, F. Two theorems of statistical separability in the perceptron. In *Symposium on the Mechanization of Thought Processes* (1959), pp. 421–456.

[93] Ruthven, I. Re-examining the potential effectiveness of interactive query expansion. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), pp. 213–220.

[94] Ruthven, I., and Lalmas, M. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review 18*, 2 (2003), 95–145.

[95] Salton, G. *Automatic information organization and retrieval*. McGraw Hill, 1968.

[96] Schapire, R., Rochery, M., Rahim, M., and Gupta, N. Incorporating prior knowledge into boosting. In *Proceedings of ICML 02: The 19th International Conference on Machine Learning* (2002), pp. 538–545.

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

[98] Sebastiani, F. Machine learning in automated text categorization. *ACM Computing Surveys 34*, 1 (2002), 1–47.

[99] Shahshahani, B.M.and Landgrebe, D.A. The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. *IEEE Transactions on Geoscience and Remote Sensing 32*, 5 (1994), 1087–1095.

[100] Shanahan, J. G., and Roma, N. Boosting support vector machines for text classification through parameter-free threshold relaxation. In *Proceedings of CIKM '03: The twelfth international conference on Information and knowledge management* (2003), pp. 247–254.

[101] Shen, X., and Zhai, C. Active feedback in ad hoc information retrieval. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), pp. 59–66.

[102] Shepard, R., Hovland, C. L., and Jenkins, H. M. Learning and memorization of classification. *Journal of Psychological Monographs 75* (1961), 1–42.

[103] Sparck-Jones, K. Notes and references on early automatic classification work. *SIGIR Forum 25*, 1 (1991), 10–17.

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

[105] Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. The perfect search engine is not enough: a study of orienteering behavior in directed search. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems* (2004), pp. 415–422.

[106] Tenenbaum, J. B. *A Bayesian Framework for Concept Learning*. PhD thesis, Massachusetts Institute of Technology, 1999.

[107] Tong, S., and Koller, D. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research 2* (2002), 45–66.

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

[109] Tong, Simon, and Chang, Edward. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia* (2001), pp. 107–118.

[110] Tzeras, K., and Hartmann, S. Automatic indexing based on bayesian inference networks. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 1993), pp. 22–35.

[111] Van Rijsbergen, C. J. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[112] Vapnik, V. N. *The Nature of Statistical Learning Theory*, 2 ed. Springer, 1995.

[113] Voorhees, E. M., and Buckland, L. P., Eds. *First Text REtrieval Conference (TREC-9)* (1992), Dept of Commerce, NIST.

[114] Voorhees, E. M., and Buckland, L. P., Eds. *Text REtrieval Conference (TREC 2005) Notebook Proceedings* (2005), Dept. of Commerce, NIST.

[115] Wu, X., and Srihari, R. 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* (2004), pp. 326–333.

[116] Yang, Y., and Liu, X. A re-examination of text categorization methods. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), pp. 42–49.

[117] Zhang, Y., Xu, W., and Callan, J. P. Exploration and exploitation in adaptive filtering based on bayesian active learning. In *Proceedings of ICML 03: The 20th International Conference on Machine Learning* (2003), pp. 896–903.