LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

UCRL-TH-230629

Measuring the Interestingness of Articles in a Limited User Environment Prospectus

R. K. Pon

May 1, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

Measuring the Interestingness of Articles in a Limited User Environment

PhD Prospectus

Raymond K. Pon

Tuesday, May 01, 2007

UCRL-TH-230629

Abstract

Search engines, such as Google, assign scores to news articles based on their relevancy to a query. However, not all relevant articles for the query may be interesting to a user. For example, if the article is old or yields little new information, the article would be uninteresting. Relevancy scores do not take into account what makes an article interesting, which would vary from user to user. Although methods such as collaborative filtering have been shown to be effective in recommendation systems, in a limited user environment there are not enough users that would make collaborative filtering effective. I present a general framework for defining and measuring the "interestingness" of articles, called iScore, incorporating user-feedback including tracking multiple topics of interest as well as finding interesting entities or phrases in a complex relationship network.

I propose and have shown the validity of the following:

1. Filtering based on only topic relevancy is insufficient for identifying interesting articles.
2. No single feature can characterize the interestingness of an article for a user. It is the combination of multiple features that yields higher quality results. For each user, these features have different degrees of usefulness for predicting interestingness.
3. Through user-feedback, a classifier can combine features to predict interestingness for the user.
4. Current evaluation corpora, such as TREC, do not capture all aspects of personalized news filtering systems necessary for system evaluation.
5. Focusing on only specific evolving user interests instead of all topics allows for more efficient resource utilization while yielding high quality recommendation results.
6. Multiple profile vectors yield significantly better results than traditional methods, such as the Rocchio algorithm, for identifying interesting articles. Additionally, the addition of tracking multiple topics as a new feature in iScore, can improve iScore's classification performance.
7. Multiple topic tracking yields better results than the best results from the last TREC adaptive filtering run.

As future work, I will address the following hypothesis: Entities and the relationship among these entities using current information extraction technology can be utilized to identify entities of interest and relationships of interest, using a scheme such as PageRank. And I will address one of the following two hypotheses:

1. By addressing the multiple reading roles that a single user may have, classification results can be improved.
2. By tailoring the operating parameters of MTT, better classification results can be achieved.

Table of Contents

1 Introduction

An explosive growth of online news has taken place in the last few years. Users are inundated with thousands of news articles, only some of which are interesting. A system to filter out uninteresting articles would aid users that need to read and analyze many news articles daily, such as financial analysts, government officials, and news reporters. Information overload is a threat to a user's ability to function, resulting in "brain-thrashing" [Denning2006], calling for a VIRT (valued information at the right time) [Hayes-Roth2006] strategy for information handling.

The most obvious approach for a VIRT strategy is to learn keywords of interest for a user [Carreira2004, Lai2003, Billsus2000]. Unfortunately, the issues related to article recommendation systems are more difficult to address than applying a simple keyword filter to weed out uninteresting articles. Although filtering articles based on keywords removes many irrelevant articles, there are still many uninteresting articles that are highly relevant to keyword searches. For example, searching for "San Francisco" in Google News will yield about 60,000 articles ordered by relevance. Unfortunately, a relevant article may not be interesting for various reasons, such as the article's age or if it discusses an event that the user has already read about in other articles.

Although it has been shown that collaborative filtering can aid in personalized recommendation systems [Wang2006] a large number of users is needed. In a limited user environment, such as a small group of analysts monitoring news events, collaborative filtering would be ineffective. To address this insufficiency to news filtering, I take a different approach by undertaking what *makes an article interesting*.

The definition of what makes an article interesting – or its "interestingness" – varies from user to user and is continually evolving, calling for adaptable user personalization. Furthermore, due to the nature of news articles, most are uninteresting since many are similar or report events outside the scope of an individual's concerns. There has been much work in news recommendation systems, but none have yet addressed the question of what makes an article interesting. In my system, iScore [Pon2007], I make the following contributions to news filtering in a limited user environment including development of the prototype system called iScore:

1. I show that filtering based on only topic relevancy is insufficient for identifying interesting articles.
2. I extract a variety of features, ranging from topic relevancy to source reputation. No single feature can characterize the interestingness of an article for a user. It is the combination of multiple features that yields 21% to 24% higher quality results. For each user, these features have different degrees of usefulness for predicting interestingness.
3. I evaluate several classifiers for combining these features to find an overall interestingness score. Through user-feedback, the classifiers find features that are useful for predicting interestingness for the user.
4. I show that current evaluation corpora, such as TREC, do not capture all aspects of personalized news filtering systems necessary for system evaluation.

Despite incorporating other article features in addition to relevancy to topics of interest, the initial version of iScore still performs poorly with users that have very general interests (as opposed to very specific interests). iScore addresses relevancy by using the output of classifiers (e.g., Rocchio) that maintain a single interest profile. Unfortunately, iScore suffers when a user has a set of interests that are orthogonal to one another, which cannot be accurately represented by a single interest profile. In this paper, I extend the initial version of iScore to address this shortcoming by extending the traditional Rocchio algorithm by using multiple profile vectors instead of one. This is a similar technique used in topic detection and tracking (TDT) [NIST2004] but applied to an online personalized news recommendation setting. Unlike in a TDT environment, where all new topics are identified and continually tracked by identifying their

related articles, identifying interesting articles for a specific user is different for two reasons: first, not all topics are of equal interest to a user; second, a user's interest in a topic continually changes overtime. A topic that may have been interesting in the past may not be interesting in the future.

Addressing these two distinctions between TDT and news recommendation and the shortfall of the existing iScore system, I make the following additional contributions through multiple topic tracking (MTT) [Pon2007a]:

1. Instead of identifying all new topics and tracking all articles for those topics as in TDT, I focus on the specific users interests, which are under continuous evolution. Focusing on only evolving user interests instead of all topics allows for more efficient resource utilization.
2. I show that the use of multiple profile vectors yield significantly better results than traditional methods, such as the Rocchio algorithm, for identifying interesting articles. Additionally, the addition of tracking multiple topics as a new feature in iScore, improves iScore classification performance by 10%.
3. For a specific user as a case study, I analyze the operating parameters for my algorithm for their resource usage and classification performance.
4. I show experimental results that multiple topic tracking yields 37.8% better results than the results from the last TREC adaptive filtering run.

In this prospectus, I discuss the general iScore framework and how MTT can aid iScore in identifying interesting articles. First, I discuss related works to illustrate that existing works do not directly address the interesting article problem and how iScore is unique. Next, I discuss the general iScore framework, detailing the currently implemented documents features that are extracted and how they are combined to generate a single "interestingness" score. Given iScore's extensibility, I then discuss MTT and how it can be added to iScore to improve performance. Initial experimental results show that iScore and MTT perform generally better than existing approaches from information retrieval. And finally, I discuss the next areas of research that I intend to tackle that will lead us closer to a solution to the interesting article problem.

2 Related Works

2.1 News Recommendation Systems

iScore is a recommendation system in a limited user environment, so the only available information is the article's content and its metadata. Work outside collaborative filtering makes use of this information in a variety of ways.

Work by [Corso2005] ranks news articles and new sources based on several properties in an online method. They claim that important news articles are clustered. They also claim that mutual reinforcement between news articles and news sources can be used for ranking, and that fresh news stories should be considered more important than old ones. In my approach, I rank news articles based on various properties in an online method, but instead of ranking articles using mutual reinforcement and article freshness, I study a different variety of features. Additionally, when training my classifiers, I also take into account that the most recent news articles are more important than older ones.

Another approach taken by [Macskassy2001] measures the interestingness of an article as the correlation between the article's content and the events that occur after the article's publication. For example, an article about a specific stock is interesting if there is a significant change in price after the article's publication. Using these prospective indicators, they can predict future interesting articles. Unfortunately, in most cases, these indicators are domain specific and are difficult to collect in advance for the online processing of new articles as they are published.

Other systems perform clustering or classification based on the article's content, computing such values as TF-IDF weights for tokens. A near neighbor text classifier [Billsus2000] uses a document vector space model. A personalized multi-document summarization and recommendation system by [Dragomir] recommends articles by suggesting articles from the same clusters in which the past interesting articles are located. I implement a variation of these methods as feature extractors in iScore. Another clustering approach, MiTAP [Damianos2003] monitors infectious disease outbreaks and other global events. Multiple information sources are captured, filtered, translated, summarized, and categorized by disease, region, information source, person, and organization. However, users must still browse through the different categories for interesting articles.

2.2 Adaptive filtering

My work in iScore is closely related to the adaptive filtering task in TREC, which is the online identification of news articles that are most relevant to a set of topics. The task is different from identifying interesting articles for a user because an article that is relevant to a topic may not necessarily be interesting. However, relevancy to a set of topics of interest is a prerequisite for interestingness. The report by [Robertson2002] summarizes the results of the last run of the TREC filtering task. In the task, topic profiles are continually updated as new articles are processed. The profiles are used to classify a document's relevancy to a topic. I discuss some of the work in this TREC task. Like much of the work in the task, I use adaptive thresholds and incremental profile updates.

In [Xu2002], the authors use a variant of the Rocchio algorithm, in which they represent documents as a vector of TF-IDF values and maintain a profile for each topic of the same dimension. The profile is adapted by adding the weighted document vector of relevant documents and by subtracting the weighted vector of irrelevant documents. Since this approach performed the best in the task, I incorporate this method in iScore. Other methods explored in TREC11 include using a second-order perceptron, an SVM, a Winnow classifier [Wu2002], language modelling [Ma2002], probabilistic models of terms and relevancy [Brouard2002], and the Okapi Basic Search System [Robertson2002a].

2.3 Ensembles

Other work, like iScore, have leveraged multiple existing techniques to build better systems for specific tasks. For example, in [Henzinger2006], the authors combine two popular webpage duplication identification methods to achieve better results. Another example is by [Lazarevic2005], which combines the results from multiple outlier detection algorithms that are applied using different sets of features.

Other work in bagging and boosting has been to identify under which conditions bagging and boosting will outperform single classifiers and how to maximize the diversity of classifiers in the ensemble. In [Esposito2004], Monte Carlo analysis is used to characterize the conditions under which the ensemble approach will outperform the single classifiers. They provided a closed form expression for the distribution of ensemble accuracy, mean, and variance. In [Melville2004], the authors also address diversity in ensembles. Classifiers that are trained on the original data and some artificial data generated from a random process that follows the training data distribution are added to the ensemble if it does not increase the ensemble training error. In [Turinsky2004], a greedy algorithm is used for selecting models in ensembles. In the greedy framework, when a new instance is to be classified, a meta-model invokes the appropriate predictive model that best corresponds to the instance. They introduce a new algorithm for learning the base models and the meta-learner for model selection, which relies on moving small amounts of data between the various data sets that are used to train the base models. Data is moved according to a simulated annealing algorithm.

A closely related ensemble work by [Yan2006] combines multiple ranking functions over the same document collection through probabilistic latent query analysis, which associates non-identical combination weights with latent classes underlying the query space. The overall ranking function is a

linear combination of the different ranking functions. They extend the overall ranking function to a finite mixture of conditional probabilistic models. I explore two methods of a linear combination approach using correlation and logistic regression, but in contrast to [Yan2006], I combine functions that are not necessarily ranking functions that can be used for ranking documents for interestingness by themselves. Each function is a different aspect of interestingness and need to be combined together to generate meaningful scores for interestingness.

2.4 Topic Detection and Tracking

Topic detection and tracking (TDT) identifies new events and groups news articles that discuss the same event. Formally, TDT consist of five separate tasks: (1) topic tracking, (2) first story detection, (3) topic detection, (4) topic linkage, and (5) story segmentation [NIST2004].

Many TDT systems, like [Allan1998], [Franz2001], and [Allan2002] are simply a modification of a single pass clustering algorithm. They compare a news story against a set of profile vectors kept in memory. If the story does not match any of the profiles by exceeding a similarity threshold, the story is flagged as a new event and a new profile is created using the document vector of the news story. Otherwise, the news story is used to update the existing profiles. Other work, such as [Makkonen2004], add simple semantics of locations, names, and temporal information to the traditional term frequency vectors used in previous work.

Although I make use of a similar single-pass clustering algorithm, there are several subtle differences between identifying interesting articles and TDT. First, not all topics are of equal interest to a user. Instead of identifying all new topics and tracking all articles for those topics as in TDT, I focus on the specific users interests, which are under continuous evolution. Additionally, I use the interestingness of topics when evaluating the interestingness of news articles that belong to their respective topics. Furthermore, a user's interest in a topic continually changes over time. A topic that may have been interesting in the past may not be interesting in the future. Consequently, I discard old profile vectors that are no longer of interest to reduce resource consumption, to speed up document evaluation, and to improve the quality of results.

2.5 Trust and Quality

Articles known to be credible may be more interesting that articles that are not. Frequent reporting errors by a news agency may contribute to a user deeming a future article uninteresting even before he reads it. To further the understanding of credibility assessment, in [Fogg2003], the authors present the prominence-interpretation theory in which two things happen when people assess credibility online: (1) the user notices something, and (2) the user makes a judgment about it. In [Nagura2006], the authors rate the credibility of news documents on the web with three metrics: commonality, numerical agreement, and objectivity. They hypothesize that as more news publishes deliver articles with similar content to the target article being assesses, the higher the credibility of the target article. Also, if numerical expressions that contradict those in other articles from different news agencies, credibility is rated lower. The credibility of the article containing subjective speculation is rated differently from those containing objective news sources. They use a dictionary approach and other heuristics to rate objectivity.

In [Pon2005], I claim that the quality of data sources can be estimated by observing the agreement among data sources and their past history of being correct. Similarly, [Roberts2006] use the value of information for evidence detection. They measure the reliability of data sources, the coherence (the agreement) among sources, and the independence between sources.

I address trust and quality by examining the objectivity of articles and the reputation of the news agency for producing interesting articles. However, addressing agreement among text documents is a difficult task and is an open area of research, which I deem to be beyond the scope of this dissertation.

2.6 Feature Selection

Since I make use of the results of my weak classifiers and the results of other feature extraction methods to build a large overall classifier, feature selection is important. But because the importance of features vary among users, is unknown a priori, and may change over time, no features can be discarded when constructing the overall classifier.

The work by [Guyon2003] is a survey on feature selection, noting cases where feature selection would improve the results of classifiers. Noise reduction and better class separation may be obtained by adding variables that are presumably redundant. Variables that are independently and identically distributed are not truly redundant. Perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them. However, very high variable correlation does not mean absence of variable complementarity. A variable that is completely useless by itself can provide a significant performance improvement when taken with others; consequently, when two variables that are useless by themselves can be useful together.

There have been three directions to variable selection: wrappers, filters, and embedded methods. Wrappers use the learning machine of interest as a black box to score subsets of variables according to their predictive power. An example of a wrapper approach is [Kohavi1997], which uses a hill-climbing approach to find a good set of features. Filters select subsets of variables as a pre-preprocessing step, independently of the chosen predictor. Embedded methods perform variable selection the process of training and are usually specific to given learning machines. To create a few baselines performance values to compare with when selecting variables for a new problem, [Guyon2003] recommend a linear predictor (e.g., linear SVM) and select variables in one of two ways: (1) a variable ranking method using correlation coefficient or mutual information; (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates.

There have been several other surveys on feature selection. [Liu2005a] calls for the integration of different feature selection algorithms. They combine the filter and the wrapper approach into a single hybrid approach for feature selection. The hybrid approach uses the independent measure of the filter approach to decide the best subsets for a given cardinality and uses the mining algorithm of the wrapper approach to select the final best subset among the best subsets across different cardinalities. Another by paper by [Liu2005] discusses feature selection applied to real-life problems. The work by [Blum1997] discusses feature-weighting methods such as Winnow [Littlestone1988]. The Winnow algorithm is very similar to that of the perceptron, except instead of additive updates, it uses multiplicative updates. Furthermore, the inputs and outputs of the Winnow algorithm are all binary. They have shown that the number of mistakes grows only logarithmically with the number of irrelevant attributes in the examples while still being computationally efficient in both time and space. [Yang1997] is a study on feature selection methods in statistical learning of text categorization. They evaluated five methods: term selection based on document frequency, information gain, mutual information, chi-square test, and term strength. They found that document frequency, information gain, and the chi-square test were most effective in their experiments. Their experiments suggest that document frequency thresholding, the simplest and lowest cost method, can be used reliably instead of the other two preferred methods.

Since decision trees use information gain to rank features, it can be considered as an embedded feature selection method. Work by [Utgoff1997] discusses an incremental decision tree algorithm that makes use of an efficient tree restructuring algorithm. However, the drawback is that any numeric data must be stored and maintained in sorted order by value and the decision tree's storage requirements will continually grow.

My current survey on feature selection has shown that existing feature selection methods, wrappers and filters are not applicable in the proposed application. I have studied the effectiveness of the embedded approach to feature selection, by constructing a new classifier which is simply the linear combination of features. I have also looked at logistic regression as a method for embedded feature selection.

2.7 Document Classification

Since news article are classified as interesting or uninteresting by iScore, it is important to note work in document classification. However, most document classification methods have been used to bin documents into particular topics, which is a different problem from binning documents by their interestingness.

There has been a lot of work in document classification in recent years. In [Wong1996], the authors classify documents, taking into account the term frequencies as well as the local relationships between available classes. They try to balance specificity, which measures the degree of precision with which the contents of a document is represented by the classification result, and exhaustivity, which measures the degree of coverage by the classification result on the domain found in a document. In [Liang2004], a SVM was used for web-page classification. In [Al-Mubaid2006], distributional clustering and a logic-based learning algorithm are used to classify documents. In [Angelova2006], the authors combine the graph/network properties of documents along with traditional content classification methods to classify documents. Work by [Diaz2006] improves classification of documents by using multiple large external corpora. This is accomplished through a mixture of relevance models.

LDA first proposed by [Blei2003] for document classification has been popular in document classification. LDA is a generative probabilistic model for collections of discrete data such as test. It is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is modeled as an infinite mixture over an underlying set of topic probabilities. The topic probabilities represent a document. Works by [Newman2006, Griffiths2004, Steyvers2006] extend [Blei2003] using LDA. They use statistical topic models to classify documents into topics not known a priori, similar to unsupervised learning methods such as traditional clustering methods. Unlike most clustering methods, the clusters of documents may share documents (i.e., overlap). In their topic model, a topic is a multinomial probability distribution over unique words in the vocabulary of the corpus. Each document is a mixture of topics and is represented as a multinomial probability vector, one probability for each topic. Given this model for a set of documents and topics, Gibbs sampling is used to estimate the topic-word and document-topic distributions.

There has also been work on boosting the performance of existing document classifiers. In [Smucker2006], the authors evaluate the effectiveness of using similarity browsing as a tool, like relevance feedback, for improving retrieval performance. They achieved performance that matched that of a traditional styled iterative relevance feedback technique.

Work by [Yu2003] classifies documents from positive and unlabelled documents; whereas, most classification schemes assume that the training data are completely labeled. They extend a support vector machine (SVM) for this task. They show that when the positive training data is not too under-sampled, their approach outperforms other methods because it exploits the natural gap between positive and negative documents in feature space. This situation is a scenario in which iScore mostly operates in, where most articles are unlabelled with a few documents that are positively labeled. Unfortunately, an SVM method is not applicable to iScore's online operating environment.

Because logistic regression is studied as a possible classifier, it is important to note other uses of logistic regression in classification. In [Genkin2004], the authors use logistic regression for text categorization to eliminate the need for ad hoc feature selection. They solve their logistic regression problem using a variation of the Gauss-Sidel method. In [Landwehr2005], the authors combine tree induction with logistic regression, where the trees' leaves contain linear regression functions.

3 The iScore System

News articles are processed in a streaming fashion, much like the document processing done in the adaptive filter task in TREC. The information about an article available to the system is the title, the name

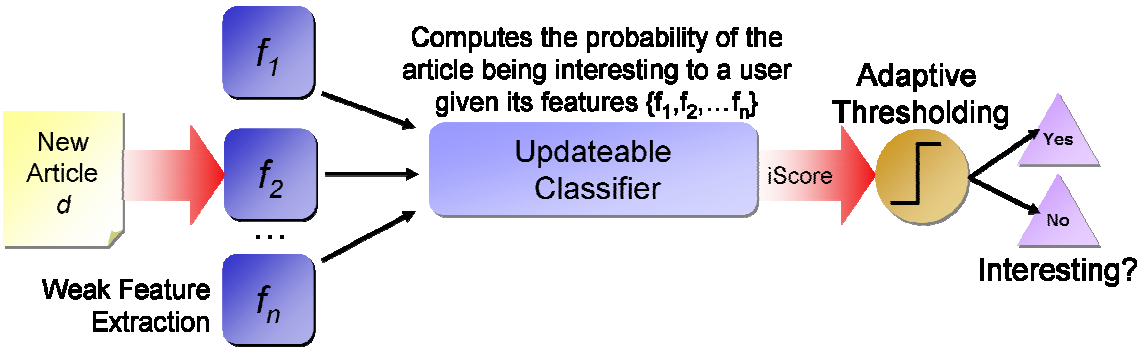


Figure 1. Article classification pipeline.

of the authors, the publication date, and the main content of the article. Articles are introduced to the system in chronological order of their publication date. Once the system classifies an article, an interestingness judgment is made available to the system by the user.

The article classification pipeline consists of four phases, shown in Figure 1. In the first phase, for an article d, a set of feature extractors generate a set of feature scores $F(d) = \{f_1(d), f_2(d), \ldots, f_n(d)\}$. Then a classifier C generates an overall classification score, or an iScore $I(d)$:

$$I(d) = C(f_1(d), f_2(d), \ldots, f_n(d)) \quad (1)$$

Next, the adaptive thresholder thresholds the iScore to generate a binary classification, indicating the interestingness of the article to the user. In the final phase, the user examines the article and provides his own binary classification of interestingness (i.e., tagging) $I'(d)$. This feedback is used to update the feature extractors, the classifier, and the thresholder. The process continues similarly for the next document in the pipeline.

3.1 Weak features for classification

In this section, I describe a set of article features that will serve as inputs into the classifier function to estimate or predict the interestingness of the article to a user. Each individual feature is a weak feature. In other words, each feature alone cannot determine the interestingness of an article for a user. The features do not capture interestingness by themselves, but are weakly correlated with an aspect of interestingness.

3.1.1 Topic relevancy

Although an article that is relevant to a topic of interest may not necessarily be interesting, relevancy to such topics is a prerequisite for interestingness for a certain class of users. I use five different methods to measure topic relevancy.

The first method is the Rocchio adaptive learning method [Rocchio1971]. Further discussion on the Rocchio algorithm is available in [Joachims1996], in which the author compares the Rocchio relevance feedback algorithm with its probabilistic variant and the standard naive Bayes classifier.

Documents are represented as a vector $\vec{d}$ in a vector space. Each dimension i of the vector space represents a token t_i. The value of the vector element is the represented token's TF-IDF value. In my experiments, tokens are stems produced by the Porter algorithm [Porter1980] Stems occurring only once in the collection are discarded to reduce the feature space, which has been shown to improve classification time and results [Yang1997].

The Rocchio algorithm maintains a profile vector $\vec{p}$ and updates it as follows:

$$\vec{p} = \vec{p} + \vec{d} \quad \text{if } d \text{ is interesting} \tag{2}$$

The relevancy score for the Rocchio algorithm of a document d is the cosine of the angle between the profile vector and the document vector.

The second method for measuring topic relevancy is a variant of Rocchio by [Xu2002], which updates profiles as follows:

$$\vec{p} = \begin{cases} \vec{p} + \alpha * \vec{d} & \text{if } d \text{ is interesting} \\ \vec{p} - \beta * \vec{d} & \text{if } d \text{ is not interesting} \\ \vec{p} - \beta' * \vec{d} & \text{otherwise and } \cos(\vec{p},\vec{d}) < t \end{cases} \tag{3}$$

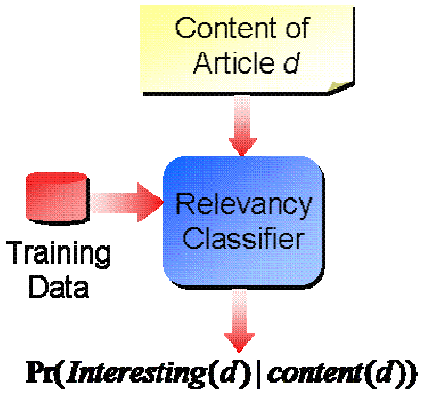


Figure 2: Topic relevancy.

The first two conditions are satisfied by user taggings. The third condition is for pseudo-negative documents, which have no taggings and its similarity with the profile is below a threshold. Good values (in TREC11) for α, β, β', and t are 1, 1.8, 1.3, and 0.6, respectively [Xu2002].

The other three methods for measuring topic relevancy use language models. An n-gram language modelling approach has been used for document classification [Peng2003], which is a method I use for finding another set of topic relevancy scores. Like naïve Bayesian classifiers, language-based modelling classifiers classify documents given the number of occurrences of grams (e.g., words or characters) in the document. Unlike naive Bayes, which assumes that grams occur independently, language modelling classifiers assume that a gram occurring is dependent upon the last $n - 1$ grams. In other words:

$$P(d) = P(g_1, g_2, ..., g_N) = \prod_{i=1}^{N} P(g_i \mid g_{i-n+1}, ..., g_{i-1}) \tag{4}$$

where N is the number of grams in the document and g_i is the i-th gram in the document d. $P(g_i|g_1,...,g_{i-1})$ can be estimated with Jelinek-Mercer smoothing[Chen1996].

In iScore, the language models are updated as new documents are processed. However, the estimation of the probabilities is time-consuming, which is addressed by compiling the models into serialized objects. However, the compilation time is proportional to the size of the models (i.e., the number of articles used to update the model), so I minimize the number of times the model is updated and compiled while still being able to produce meaningful results. I compile the models at regular intervals (i.e., every time there is an update to the model and on a daily basis). To avoid biasing the models from classifying articles as uninteresting (since there are an overwhelming number uninteresting articles compared to interesting ones) and to reduce compilation time, I update the models with all interesting articles, and only update the models with uninteresting articles if the number of uninteresting articles already used to update the model is less than the number of interesting article seen.

Using language models, I extract three topic relevancy measurements for each document. The first measurement is $P(\text{Int} \mid d)$, using a 6-gram character model. Another measurement is $P(\text{Int} \mid d)$, using a uni-gram model where grams are tokens consisting of two words – equivalent to a naive Bayesian classifier. The final measurement is $\log(P(\text{Int}, d))$, which is the sample cross-entropy rate between the language model of interesting past articles and the current article, using a 6-gram character model.

3.1.2 Uniqueness

Articles that yield little new information compared to articles already seen may not be interesting. In contrast, an article that first breaks a news event may be interesting. Anomalous articles that describe a

rare news event may also be interesting. For example, in [Rattigan2005], interesting articles may be produced by rare collaborations among authors. Methods for outlier detection include using mixture models [Eskin2000], generating solving sets [Angiulli2005] and using k-d trees [Chaudhary2002], to identify outliers. Other more recent work by [Pilla2005] proposes a new statistic based on a score process for determining the statistical significance of a putative signal that may be a small perturbation to a noisy experimental background. Work in network security, such as intrusion detection has already leveraged work in outlier detection. For example, [Liao2002] uses kNN for intrusion detection. The occurrence of system calls is used to characterize program behavior. A system call is treated as a word in a long document and the set of system calls generated by a process is treated as the "document."

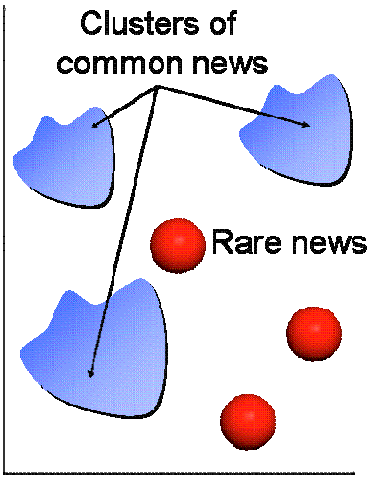


Figure 3: Uniqueness

The first anomaly measurement I use is the dissimilarity of the current article with clusters of past articles. Each document is represented as a document vector, as in the Rocchio algorithm. I maintain at most *maxCluster* clusters, which are also represented by vectors. I also maintain a count of documents that each cluster contains. The anomaly score is the weighted average dissimilarity score between the current document and each cluster, weighted by each respective cluster's size (i.e., number of contained documents):

$$f_{Cluster-Anomaly}(d) = 1.0 - \frac{\sum_{p \in P} \cos(\vec{d}, \vec{p}) size(p)}{\sum_{p \in P} size(p)} \quad (5)$$

After the article has been evaluated, I update the clusters. If the similarity between an article and a cluster is above a threshold, then the article is added to the cluster. An article may belong to more than one cluster. If there are no clusters to which the document is similar to, then a new cluster is added the list of clusters given the document's vector. If there are already *maxClusters* clusters, the cluster that has been updated least is discarded and a new cluster is added in its place. The least used clusters are tracked by maintaining an ordered list of clusters where the last cluster in the list has been most recently updated.

The threshold is also progressively updated. When there have been few documents seen so far, the threshold is set low to encourage document clustering since the cluster sizes are small at the start of collection processing. As more documents are seen, the clusters are large enough such that I can accurately identify outliers, and so the threshold is incremented by *growthRate* (reaching a maximum threshold) whenever no new clusters have been added. In my experiments, I set the maximum threshold, *growthRate*, the initial threshold, and *maxCluster* to 0.5, 0.01, 0.1, and 200, respectively.

Two other methods for anomaly detection use language models. In the first model, I maintain compiled models trained on the documents already seen, estimating the following:

$$f_{LM-Anomaly}(d) = \log(P(d \mid \text{documents seen before})) \quad (6)$$

I experiment with a 6-gram character model, and a bi-gram model, where grams are word stems.

The second language model-based anomaly detection method measures the significance and the presence of new phrases. I maintain a background model of all the documents previously seen and compare it with the language model of the current document. I measure the sum of the significance of the degree to which phrase counts in the document model exceed their expected counts in the background model. I consider only the top-10 phrases that exceed their expected counts. I use a tri-gram model where grams are word tokens.

Because language models are costly to compile, I compile the models in increasing intervals. Each time a

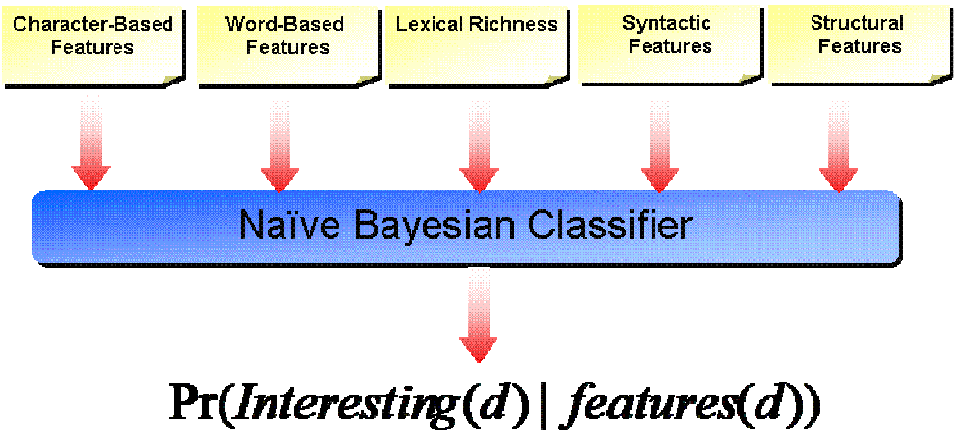


Figure 4: Writing style

language model is compiled, the next recompile is scheduled to occur after seeing the next $x + 1$ documents, where x is the number of documents seen before the current compile time. This increasing interval scheduling allows for language models to be updated and compiled frequently when there have been few documents seen. But after seeing many documents, language models should not change much unless there is a significant change in the contents of the articles seen, so the recompile intervals are increased as more documents are seen, capping off at 10,000 documents for the recompile interval.

3.1.3 Source reputation

Source reputation estimates an article's interestingness given the source's past history in producing interesting articles. Articles from a source known to produce interesting articles tend to be more interesting than articles from less-reputable sources. Moreover, specific sources may specialize in particular topics in which the user is interested. A news article's source may be its news agency or its author. In my experiments, I use the article's author(s). I estimate the article's source reputation score as the average proportion of documents produced by the authors that were interesting in the past:

$$f_{Source-Rep}(d) = \frac{\sum_{a \in authors(d)} \frac{\#\text{ Int articles written by } a}{\#\text{ Articles written by } a}}{|authors(d)|} \tag{7}$$

3.1.4 Writing style

Most work using the writing style of articles has mainly been for authorship attribution of news articles [Li2006] and blogs [Koppel2006]. Other than authorship attribution, changes in linguistic features over the course of a document have been used to segment documents as well [Chase2006]. Instead of author attribution and document segmentation, I use the same writing style features to infer interestingness. For example, the vocabulary richness [Tweedie1998] of an article should suit the user's understanding of the topic (e.g., a layman versus an expert). Also writing style features may help with author attribution, which can be used for classifying interestingness, where such information is unavailable.

I use a naïve Bayesian classifier trained on a subset of the features from [Chesley2006], including syntactic, structural, lexical, word-based, and vocabulary richness features. Like the language models used in the topic relevancy measurements, I balance the number of positive and negative articles used to update the classifier. The writing style score measured is:

$$f_{Writing-Style}(d) = P(\text{Int} \mid writingStyleFeatures(d)) \tag{8}$$

3.1.5 Freshness

Generally, articles about the same event are published around the time the event has occurred. This may also be the case for interesting events, and consequently interesting articles, so I measure the temporal distance between the last k interesting articles and the current article:

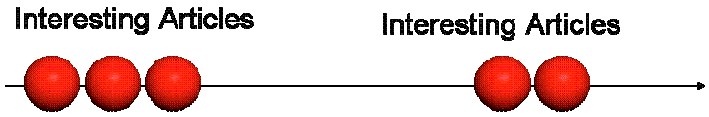


Figure 5: Freshness

$$f_{Freshness}(d) = \frac{1}{k} \sum_{d' \in \text{last } k \text{ Int articles}} \log\left(Time(d) - Time(d') + 1\right) \quad (9)$$

I measure the log of the temporal distance between an interesting article and the current article since I are interested in the order of magnitude in time differences. For example, an article published one day after the last interesting article should be significantly more interesting than an article published 100 days after the last interesting article. On the other hand, two articles published long after the last interesting article should be approximately equally old, with respect to the last interesting article, even though they may have been published 1000 and 1500 days, respectively, after the last interesting article.

3.1.6 Subjectivity and polarity

The sentiment of an article may also contribute to a user's definition of interestingness. For example, "bad news" may be more interesting than "good news" (i.e., the polarity of the article). Or, subjective articles may be more interesting than objective articles. Polarity identification has been done with a dictionary [Mishne2005] and blog-specific features [Wiebe2000]. Others have looked at subjectivity tagging, using various NLP techniques [Wiebe2004]. The density of subjectivity clues in the surrounding context of a word has been used to infer its subjectivity [Wiebe2002] as well.

I measure four different features of this feature class: polarity, subjectivity, objective speech events, and subjective speech events. A speech event is a statement made by a person, such as a quotation. Using the MPQA corpus [Wiebe2002] to train 6-gram character language model classifiers, I classify each sentence in the document to determine its polarity, subjectivity, and the presence of objective or subjective speech events. The MPQA corpus is a new article collection from a variety of news sources annotated for opinions and other states, such as beliefs, emotions, sentiments, and speculations. For each document and each feature in this feature set, I measure:

$$f_{class}(d) = \frac{1}{|sentences(d)|} \sum_{s \in sentences(d)} P(class \mid s) \quad (10)$$

where *class* is whether the sentence has negative polarity (i.e., bad news), the sentence contains subjective content (i.e., opinions, speculation), the sentence contains an objective speech event, or the sentence contains a subjective speech event.

3.2 Classification

The overall classifier computes the final iScore given all the features values generated by the feature extractors. Because the features are continually refined as more documents are seen, some of the feature values may be erroneous for early documents. Also, not all the features may be useful in predicting interesting articles for a user, depending on the user's criterions. The addition of useless features has been shown to degrade the performance of classifiers [Forman2004]. Consequently, an overall classifier must be incrementally updateable, robust against noisy and potentially useless features, and generate meaningful final scores for interestingness. I evaluate four classes of classifiers: a naive Bayesian classifier, non-incremental classifiers using a sliding window, temporal inductive transfer classifiers, and a linear combination using correlation for weights.

3.2.1 Naïve Bayesian classifier

A naive Bayesian classifier is a simple yet popular method for classification. The classifier assumes that each feature from the set of features F is independent given the class of the document, or its interestingness. Using Bayes' rule and the independence assumption, I find:

$$I(d) = P(\text{Int} \mid F(d)) \approx \frac{P(Int)\prod_f P(f(d) \mid \text{Int})}{P(F(d))} \tag{11}$$

The probabilities can be estimated by maintaining statistics over feature values using kernel estimators [John1995].

3.2.2 Non-incremental classifiers

I evaluate three classifiers that are robust against irrelevant features, but are not incrementally updateable, so these classifiers are trained on a sliding window of documents. Unaltered, for the classifiers to be continually trained, all the documents' features and their taggings would have to be stored, and each classifier would have to be rebuilt each time a document is processed, making this approach infeasible.

Since recent articles are more useful in predicting interestingness than older ones, I build windowing classifiers such that the classifiers are trained on only the last M interesting documents and the last N uninteresting documents. And the classifiers are rebuilt on an increasing interval schedule, like the compilation schedule for the language models used in anomaly detection. In my experiments, the maximum number of documents in between rebuilds of the classifier is 300 documents, and the maximum numbers of positive and negative documents in a window are both 500 documents. The interval growth rate is two documents.

In this windowing approach, I first evaluate the C4.5 decision tree, built by the J48 algorithm [Quinlan1993]. A tree is generated using the information gain of each feature, with features with high information gain at the top of the tree and features with low information gain at the bottom of the tree. The tree is then pruned to remove branches that have low confidence in their predictive abilities; making it robust against irrelevant features.

The second classifier uses logistic regression, which models the posterior probability of interestingness as a logistic function on a linear combination of features:

$$I(d) = P(\text{Int} \mid F(d)) = 1 / \left(1 + e^{-\sum_{f \in F} \lambda_f f(d)} \right) \tag{12}$$

A similar approach is taken in [Cessie1992] to combine multiple ranking methods. A quasi-Newton method and ridge estimators are used to search for optimal values for λ_f [Cessie1992].

In my experiments, I find that logistic regression is more accurate than C4.5 under the windowing scheme, so I evaluate logistic regression with bagging [Breiman1996]. Bagging mitigates the instability of learning methods by building an ensemble of classifiers trained on randomly sampled instances from the training data. In my experiments, I build 100 ensemble classifiers.

3.2.3 Tix

I modify a method used to address concept drift, called Temporal Inductive Transfer, or Tix [Forman2006]. For every M articles processed, a new classifier is built using a base induction algorithm. The input feature vector consists of the values generated by the feature extractors along with P additional binary features. The P features are generated by predictions that the P previous classifiers would have made for the current article. To bootstrap the Tix process, the first M articles (articles in the first interval) are processed by a classifier that is continually rebuilt as new documents are read. After the first interval, the regular Tix procedure begins. In my experiments, I use logistic regression as my base induction algorithm, P = 128 classifiers, and M = 1000 articles.

3.2.4 Linear correlator

I also evaluate a linear correlator classifier that uses the correlation between a feature and interestingness. Intuitively, if a feature is highly correlated with interestingness, it should be weighted more in classifying the document. Unfortunately, it is assumed that each feature is independent, ignoring the possibility that two features that perform poorly alone in predicting interestingness may perform well when combined together [Guyon2003].

As each document is processed, I incrementally compute the Pearson's correlation $corr_f$ of each feature f with interestingness. The classifier calculates an iScore as follows, weighting each feature with its interestingness correlation:

$$I(d) = \frac{\sum_f corr_f \sigma(f(d))}{\sum_f corr_f}, \quad \sigma(f(d)) = \frac{1}{1 + e^{-a_f (f(d) - t_f)}} \tag{13}$$

where $\sigma(f(d))$ is the sigmoid function. Because each feature value is a real number, not necessarily bounded between 0 and 1 and the final iScore value is a real number between 0 and 1, the sigmoid function is used to squeeze $f(d)$ to such a value.

The parameter t_f is the threshold of the sigmoid function. If $f(d)$ is less than t_f, the sigmoid function approaches 0. For $f(d)$ greater than t_f, the sigmoid function approaches 1. I assume that feature values belong to two different normal distributions, one for interesting articles and one for uninteresting articles. I incrementally maintain the averages and standard deviations for both distributions. If the feature is directly correlated to interestingness, the average feature value of interesting articles is greater than that of uninteresting articles. I compute the predicted true positive and true negative rates, given the cumulative distribution functions of the two distributions, for any threshold for a feature. And so for each threshold (according to some granularity) between the two averages, I compute a utility measure, and select the threshold with the greatest utility. In my experiments, I use TREC's T11SU for the utility measure. In the case where there are ties in utility, the threshold closest to the mid-point between the averages of feature values of interesting and uninteresting articles is selected. For features inversely correlated to interestingness, the slope of the sigmoid function is negated and the computations for the accuracy rates are adjusted accordingly.

The parameter a_f is the slope of the sigmoid function, which determines how step-like the sigmoid function is. If the lone feature is able to predict interestingness by simply thresholding, the sigmoid function should be more step-like and it should generate 0's and 1's with clear certainty. On the other hand, if the feature is poor at predicting interestingness, the feature should be less step-like, generating more ambiguous scores. And so I use the threshold's utility measure, which is proportional to the feature's predictive power, for the slope.

3.3 Adaptive thresholding

After the overall classifier has generated an iScore, the iScore is thresholded to classify the document's interestingness. Instead of using a static threshold, I dynamically adjust the threshold in a similar fashion as the threshold computation for the linear correlator, with a few modifications. Because iScores are real numbers bounded between 0 and 1, I can evaluate the efficacy of every threshold between 0 and 1 in increments of 0.01 and do not have to assume that interesting and uninteresting articles are normally distributed. And in the case of ties between the utility measures, I select the threshold that has deviated least from the previous threshold computed for the last document. The utility measures I evaluate are T11SU and F-measure f_β, where $\beta = 0.5$.

4 Multiple Topic Tracking

4.1 Motivation

Many information filtering algorithms are based on the Rocchio algorithm, which represents topics and documents as vectors. Each value of the vector is a TF-IDF value for its respective term [Rocchio1971]. A single profile vector p is maintained. For each document, the cosine similarity, or the cosine of the angle between the document vector d and the profile vector is measured.

$$\cos(d, p) = \frac{d \bullet p}{|d||p|} \tag{14}$$

The document is classified as relevant or interesting if the similarity is greater than some threshold. The profile vector is updated by adding the vector of interesting documents to the profile vector. There are variations of the Rocchio algorithm, such as subtracting irrelevant document vectors from the profile vector [Xu2002].

The Rocchio algorithm tries to find the single ideal query, or vector, that would find all interesting articles, by using the centroid of the cluster that would contain all interesting articles. However, because of the diversity in the set of interests just for a single user, finding a single ideal query is not possible [Schapire1998]. If a user has a wide range of interests, using one vector to represent his interests would dilute the sensitivity of the Rocchio algorithm. Figure 6 illustrates this problem. Although the cluster of all the interesting documents would contain interesting documents, it would also contain many uninteresting articles due to its size. If the user is interested in many orthogonal topics, then the encompassing cluster would be much larger and would also contain many more uninteresting articles as well.

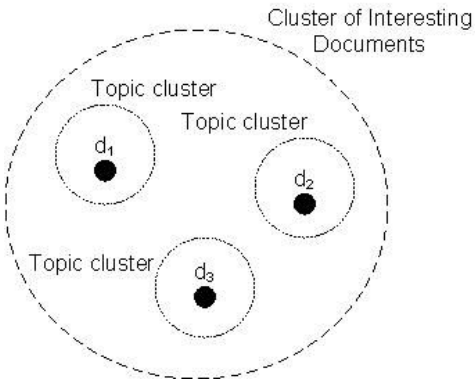


Figure 6: Failure of identifying relevant documents for multiple topics.

Instead, in MTT, a set of more narrow queries or profile vectors that more accurately represent a user's interests than a single vector is maintained. For example, in Figure 6, MTT maintains smaller topic clusters instead of the larger encompassing cluster, improving classification precision. In other words, a set of experts is generated and maintained (one for each specific interesting topic) instead of referring to a single general expert. Using specialized profiles instead of a single general profile reduces classification bias by focusing more on specific topics; at the same time, using multiple vectors keeps classification variance low.

Also the traditional Rocchio algorithm and TDT algorithms do not take into account the different degrees of interest among different topics. By focusing on individual topics, MTT can learn the user's level of interest for a specific topic and relate the topic's interestingness to related articles; thereby, improving the quality of news recommendation results. By associating a level of interest for specific topics, MTT can also learn when a user's interests have changed. Topics that were of interest in the past may no longer be interesting in the future. Topics that have grown to be uninteresting to the user can be discarded.

4.2 Algorithm

In MTT, each document and profile vector is represented as a TF-IDF vector, where each value of the vector is the TF-IDF value of the vector element's corresponding stemmed term. Terms are stemmed

using the Porter algorithm [Porter1980] and stop-words are ignored. A set of profiles P is maintained, which is initially empty. Until an interesting article arrives on the document stream, each article is scored with a 0. When an interesting article does arrive, a new profile vector p_1 is created using the article's TF-IDF vector and added to P. Each subsequent article on the document stream with a document vector d is processed as follows:

1. Find the profile vector with the maximum similarity with d. This profile represents the closest topic of interest to the document and is denoted as p_{max}.
2. The score for a document is the product of the precision of p_{max} for predicting interesting articles and the similarity between p_{max} and d. In other words:

$$f_{MTT}(d) = precision(p_{\max}) * \cos(p_{\max}, d) \tag{15}$$

   The precision of p_{max} describes how well p_{max} can accurately identify interesting articles. The precision describes how interesting the user finds the topic that the profile vector represents. By multiplying the interestingness of the topic with the document's similarity to the topic, I relate the interestingness of the containing topic to the document.

3. If the article is interesting and the similarity between d and p_{max} is less than the cluster threshold $t_{cluster}$, a new profile is generated using d. However, if the similarity is greater than or equal to $t_{cluster}$, then p_{max} is updated as follows:

$$p_{\max} = p_{\max} + d \tag{16}$$

   Intuitively, a new profile is created because a new topic has been encountered. Each profile vector is simply the centroid of the cluster of its related articles.

4. If the article is not interesting and the similarity between d and p_{max} is greater than the classification threshold $t_{classification}$, then p_{max} is updated as follows:

$$p_{\max} = p_{\max} - \gamma * d \tag{17}$$

   Because the profile misclassifies the article as interesting, the cluster is updated to remove the influence of terms that are not useful for predicting interestingness. This technique is similar to query zoning [Singhal1997], where a select set of non-relevant articles that have some relationship to a user's interests is used for updating profile vectors. The parameter γ determines how much weight negative documents in the query zone have on the topic profile.

As more documents are processed, it is possible that many profiles may be kept and maintained, making MTT expensive. However, there are two discard methods that can reduce resource consumption and improve the quality of results. The first method discards profiles whose topics are no longer interesting. Mentioned earlier, each profile vector has an associated precision for identifying interesting articles, which is defined as:

$$precision(p) = \frac{\#\text{Interesting articles w/ } \cos(p,d) > t_{classification}}{\#\text{Articles w/ } \cos(p,d) > t_{classification}} \tag{18}$$

In other words, the precision of a profile p is the proportion of articles that belong to p that are truly interesting. Profiles that have a precision less than the threshold $t_{precision}$, are discarded because the topic that the profile represents is no longer interesting to the user.

The second method discards profiles whose topics are no longer active or current. At most M profiles are maintained in memory. If there are already M profiles being maintained, when a new profile must be created, then an old profile must be discarded and the least recently used profile is selected for discard. A profile is considered "used" when an interesting article best matches the profile (i.e., when the profile is selected as p_{max}).

5 Experimental Results

iScore is implemented with an assortment of tools in Java. The system pipeline is implemented with the IBM UIMA framework [IBM2006]. LingPipe [Alias-I2006] is used for building language models and related classifiers. OpenNLP [OpenNLP2006] is used for sentence detection. Other classifiers are from Weka [Witten2004].

I evaluate the iScore framework described in section 3 against two data sets. The first data set is a collection of 35,256 news articles from all Yahoo! News RSS feeds [Yahoo2007], collected between June and August 2006. The classification task is to identify which articles come from which RSS feed. RSS feeds considered for labeling are feeds of the form: "Top Stories <*category*>", "Most Viewed <*category*>", "Most Emailed <*category*>", and "Most Highly Rated <*category*>." Because user evaluation is difficult to collect and such data is often sparse, the Yahoo! news articles and their source feeds are used for their resemblance to user labeled articles. For example, RSS feeds such as "Most Viewed Technology" is a good proxy of what the most interesting articles are for technologists. Other categories, such as "Top Stories Politics," are a collection of news stories that the Yahoo! political news editors deem to be of interest to their audience, so the feed also would serve well as a proxy for interestingness.

The other data set comes from the TREC11 adaptive filter task, which uses the Reuters RCV1 corpus and a set of assessor manual taggings for 50 topics, such as "Economic Espionage." The corpus is a collection of 723,432 news articles from 1996 to 1997. Although the TREC adaptive filter work addresses topic relevancy and not necessarily interestingness, the task is done in a similar online and adaptive fashion as in iScore, and the topics may be reasonable proxies for a set of users.

I also evaluate the multiple tracking feature on one additional data set. The data set consists of user taggings and articles from the web collected between August 2006 to January 2007. Users are asked to tag articles that they read as interesting or not interesting using a web browser plug-in. Web pages of the referring page of the tagged article are also downloaded to determine the articles that the user chose not read, which is used to infer uninterestingness for the user. After manually discarding junk web pages (i.e., non-news articles), a total of 13,281 web pages remain with six users who tagged at least 49 interesting articles. Using this data set, the classification task is to identify the interesting news articles for each of the six users from each user's own pool of articles that he had access to. Unfortunately, this data set is small compared to the other data sets, but it should provide some insight on the relative performance of classifiers on real-world data.

I use precision, recall, and F-measure, where β = 0.5, which weights precision more than recall, for system evaluation:

$$F_\beta = \frac{\left(1+\frac{\beta}{2}\right)|\text{Int Articles Retr}|}{|\text{Articles Retr}|+\frac{\beta}{2}|\text{Int Articles}|} \tag{19}$$

F-measure is 0 when the number of articles retrieved is 0. TREC11's T11SU is also used for comparing the performance of iScore with the work done in TREC11:

$$T11SU = 2*\max(T11NU, 0.5)-1$$
$$T11NU = \frac{2*|\text{Int Articles Retr}|-|\text{Unint Articles Retr}|}{2*|\text{Interesting Articles}|} \tag{20}$$

For systems that retrieve no articles, the system would have a T11SU score of 0.33.

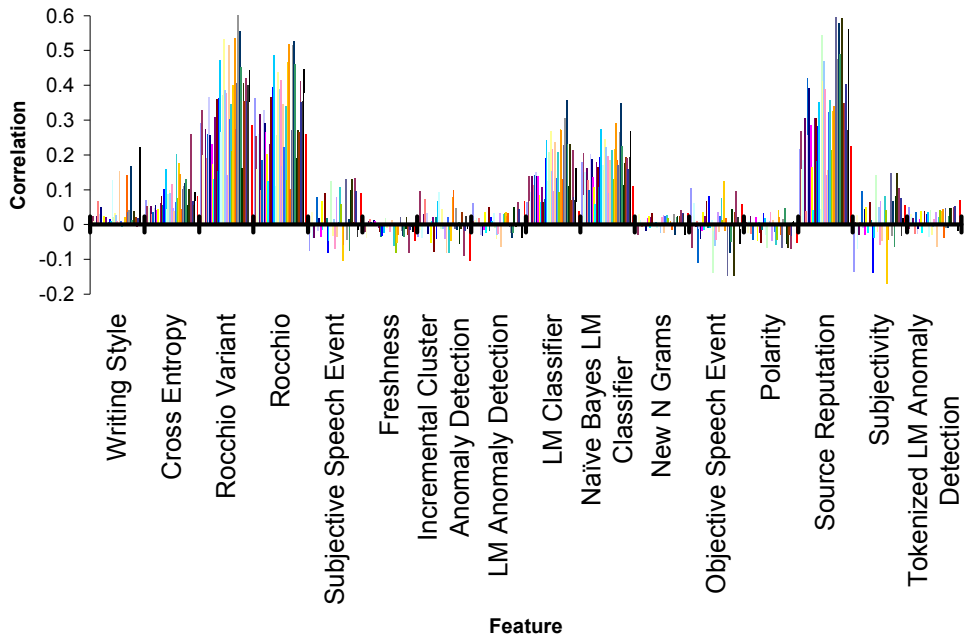


Figure 7: Correlation of features with interestingness of the Yahoo! RSS Feed articles. Each vertical line represents an RSS feed.

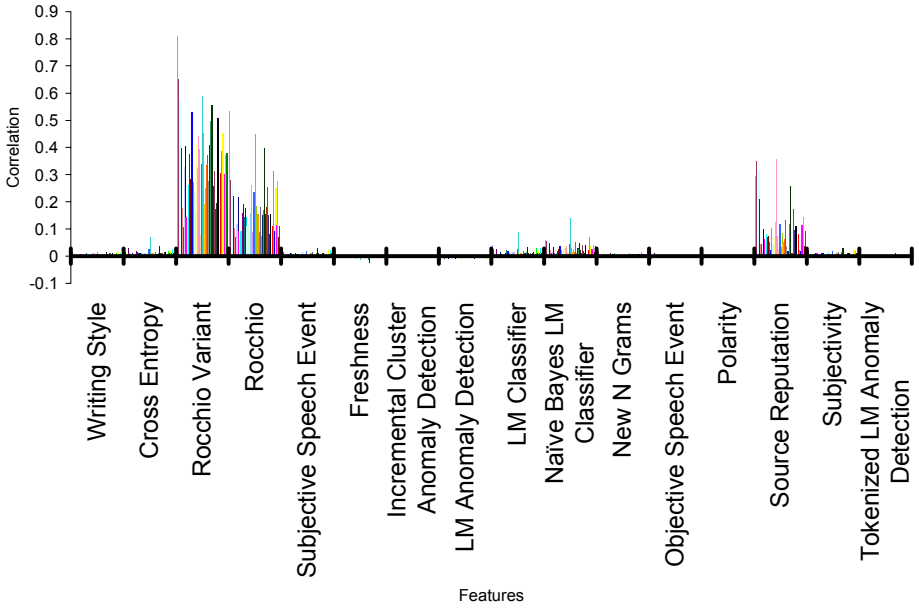


Figure 8: Correlation of features with interestingness of the TREC11 articles and tags. Each vertical line represents a topic.

Resource consumption is measured as the number of profiles that is currently being maintained. The run-time and memory consumption of MTT is linear with respect to the number of profiles. Statistical significance tests are applied where appropriate using the t-test at $p \leq 0.1$.

5.1 iScore System Framework Evaluation

5.1.1 Feature analysis

Figure 7 shows the correlation of the features with interestingness in each of the RSS feeds. For most feeds, the topic relevancy and source reputation features are significantly directly correlated with interestingness. Other features, such as writing style, speech events, anomaly detection, and subjectivity have varying correlation magnitudes and directions with interestingness, depending on the RSS feed. The RSS feeds capture a variety of criterions that users may use when evaluating the interestingness of an article.

On the other hand, Figure 8 shows that the topic relevancy and source reputation scores are the only features correlated with relevancy in the adaptive filter task. As expected, the Rocchio variant is the most correlated feature since it was the best performing filter in TREC11. Although the figure shows that the adaptive filter task captures topic relevancy well, topic relevancy is only a prerequisite for interestingness and is not sufficient for an article to be interesting. The TREC11 taggings and articles do not capture other aspects of interestingness well.

5.1.2 Overall performance

For the Yahoo! RSS articles, I evaluate the performance of each overall classifier against several well known topic relevancy classifiers: Rocchio, the Rocchio variant, and the 6-gram character language modelling classifier. Each classifier is coupled with the adaptive thresholding mechanism, using f_β as the utility metric.

Figure 9 shows the overall performance of the iScore classifiers compared to the baseline classifiers. The averages across RSS feeds, of precision, recall, and f_β are plotted in the graph. iScore with naïve Bayes outperforms the best baseline classifier (the language modelling classifier) by 21% in terms of f_β. iScore with the linear correlator, logistic regression, and logistic regression with bagging also perform as well as most of the baseline classifiers. iScore classifiers using Tix and the decision tree under the windowing scheme in iScore perform the worse. However, the decision tree yields high recall at the cost of precision.

Although naïve Bayes in iScore outperforms all the other classifiers used in iScore, it has a slight advantage. Naïve Bayes can be incrementally updated quickly. The other classifiers can not be updated as every new document is processed due to computational and storage costs. The windowed classifiers have to operate over a sliding window of data items and are rebuilt at increasing intervals. Consequently, the windowed classifiers are trained with less data and are not necessarily up-to-date with the information about the last document processed.

Figure 11 shows the performance of iScore using naïve Bayes over each of the individual feeds along with the number of articles in each feed. The feed with the worse results is the "Highest Rated Travel" feed due to the low number of articles in the feed. However, there are feeds that performed poorly despite the high number of articles in those feeds. Feeds, such as "Highest Rated" contain a variety of articles from different topics, so the topic relevancy measures, which are the most highly correlated features with interestingness overall, do not work well for these feeds.

Since iScore uses some of the methods designed for the TREC task and topic relevancy is a prerequisite for interestingness, I compare the iScore classifiers (coupled with the adaptive thresholder optimized for T11SU) with the best filters from each participating group in TREC11 in Figure 10. Although iScore did not perform as well as the best filter, iScore with the linear correlator did perform generally well compared with most of the other filters. The next best iScore classifier is the naïve Bayesian classifier, which is consistent with the Yahoo! data set. Logistic regression, logistic regression with bagging, Tix, and C4.5 yield the worst precision and f_β score but high recall.

5.1.3 Performance over time Periods

Figure 12 shows the performance of each classifier over different time periods using the Yahoo! RSS articles. Each time period contains 5000 articles. The best classifier used for iScore is the naïve Bayesian classifier, outperforming the best baseline classifier (the language modelling classifier) by 24.3% on average. iScore using the naïve Bayesian classifier only performs worse than the baseline classifiers in the first time period. Because iScore has three layers of learning that must be done (i.e., the feature extractors, the overall classifier, and the adaptive thresholding); whereas, the baseline classifiers only have two layers (i.e., the classifier itself and the adaptive thresholding), iScore performs poorly at first due to

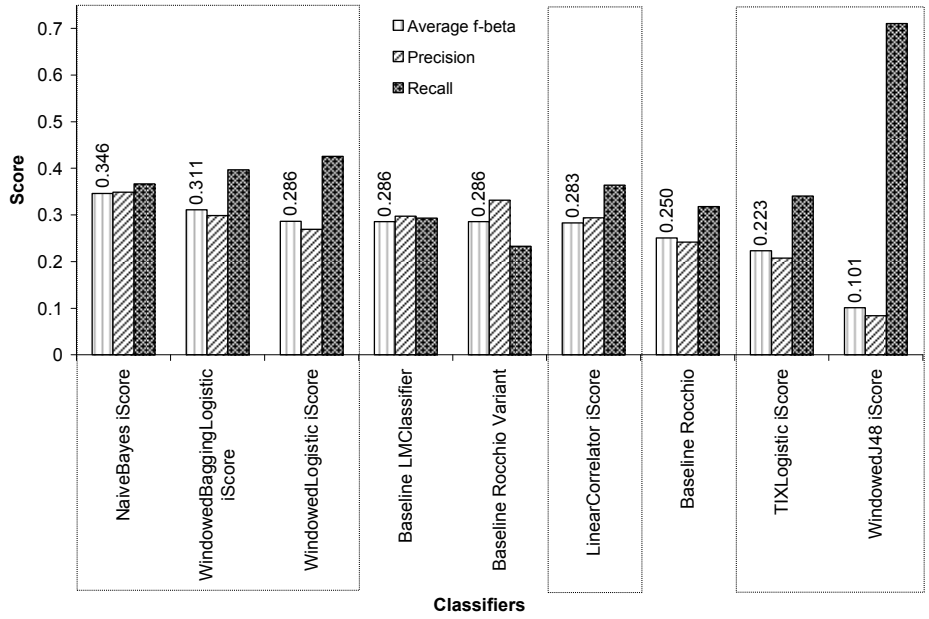


Figure 9: Overall performance of classifiers over the Yahoo! RSS articles. The iScore classifiers are outlined.

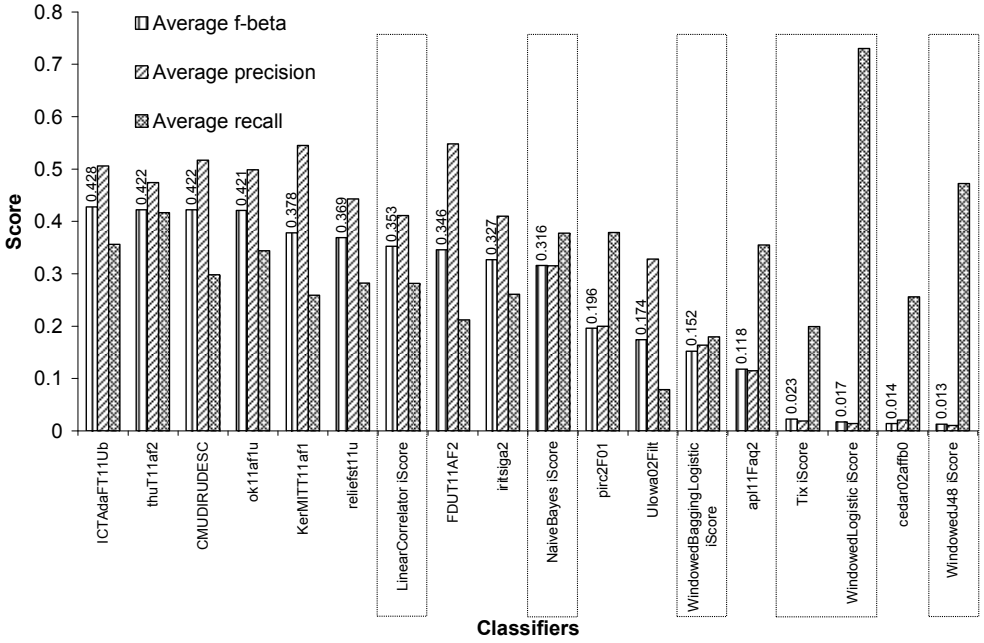


Figure 10: Overall performance of classifiers over the TREC articles. The iScore classifiers are outlined.

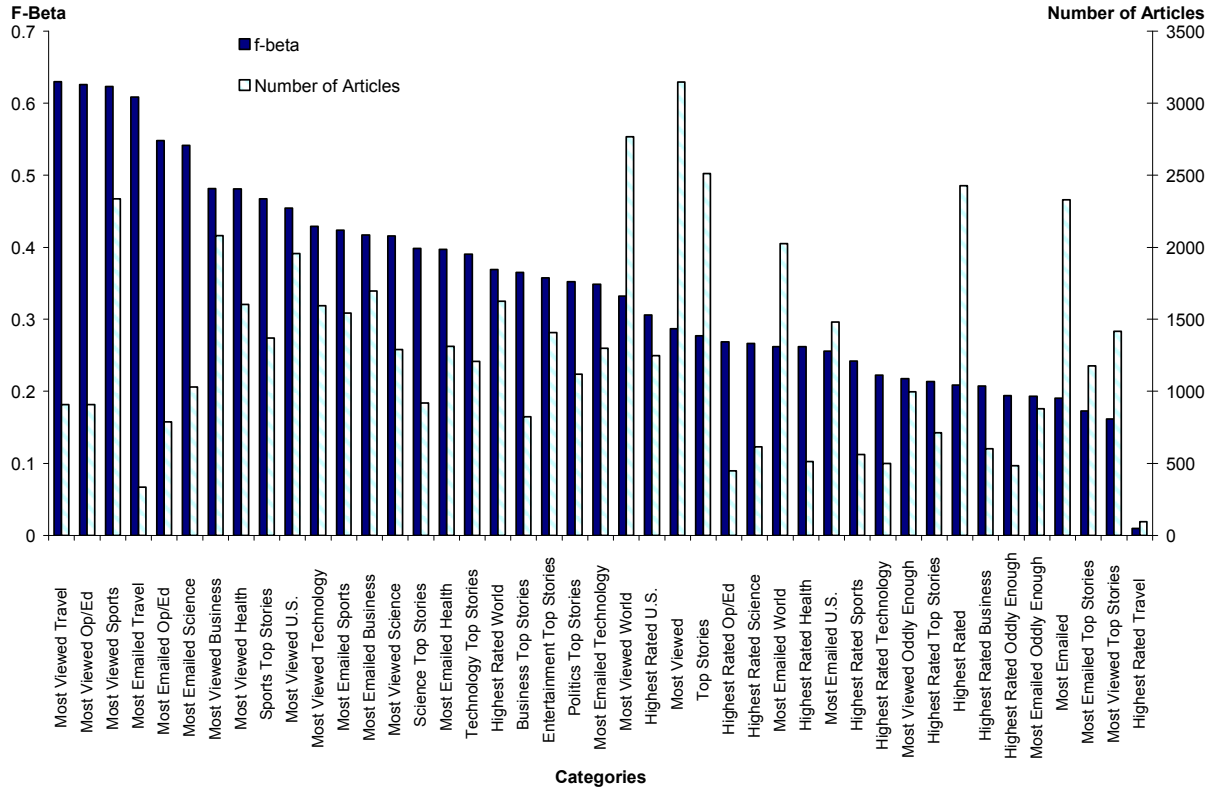


Figure 11: Performance of iScore (using Naive Bayes) in individual categories along with the number of articles in each category.

propagation error among the layers. The other iScore classifiers perform generally better than the baseline classifiers with the exception of the decision tree, which fails to improve as more documents are processed.

The dip in performance in the sixth time period by most classifiers is due to concept drift introduced by a pause in the collection of new articles. Logistic regression and logistic regression with bagging are the most affected by the drift. Also, Tix, which is intended to address concept drift, is interestingly also affected by the pause in data collection.

I also compare the T11SU performance of iScore with the best filters from TREC11 over time in Figure 13. Each time interval is a month's worth of articles. As in the overall performance analysis of iScore, logistic regression, logistic regression with bagging, Tix, and the decision tree perform poorly; whereas, naïve Bayes and the linear correlator perform well. In the beginning periods, naïve Bayes performs poorly compared to the TREC filters and the linear correlator, but in the latter periods, it outperforms them. The lack of overall improvement in Figure 10 by iScore over the TREC filters and the slow increase in performance in Figure 13 are attributed to the additional learning layers in iScore. Also the multitude of useless features for the TREC task, as shown in Figure 8, is a contributing factor since iScore must spend more time learning which features are irrelevant. These problems can be addressed with more training, but because there are few relevant documents for each TREC topic distributed sparsely across the entire collection (proportionally much less than in the Yahoo! collection), iScore cannot immediately learn enough to outperform the other classifiers until it has seen more articles.

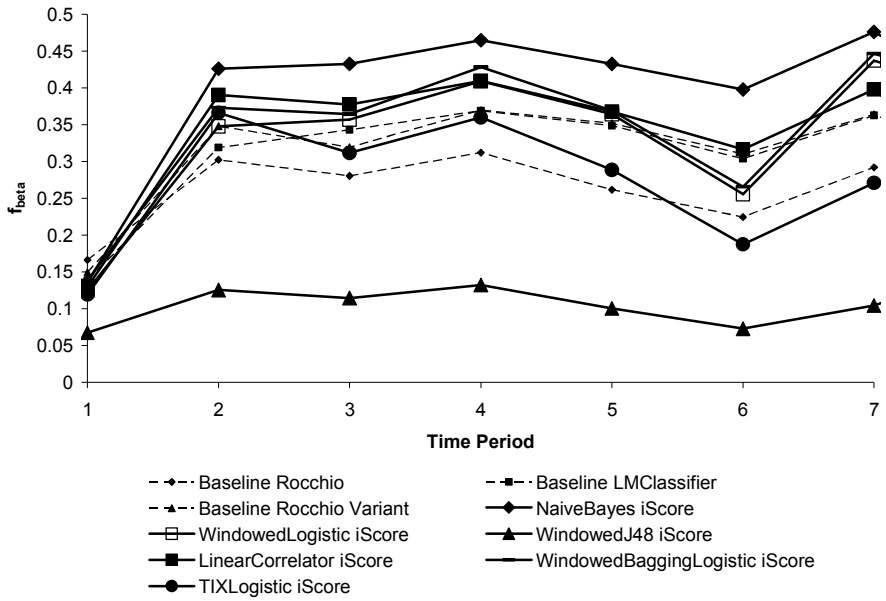


Figure 12: Performance of classifiers over time periods over the Yahoo! RSS articles. Each time period contains 5000 articles.

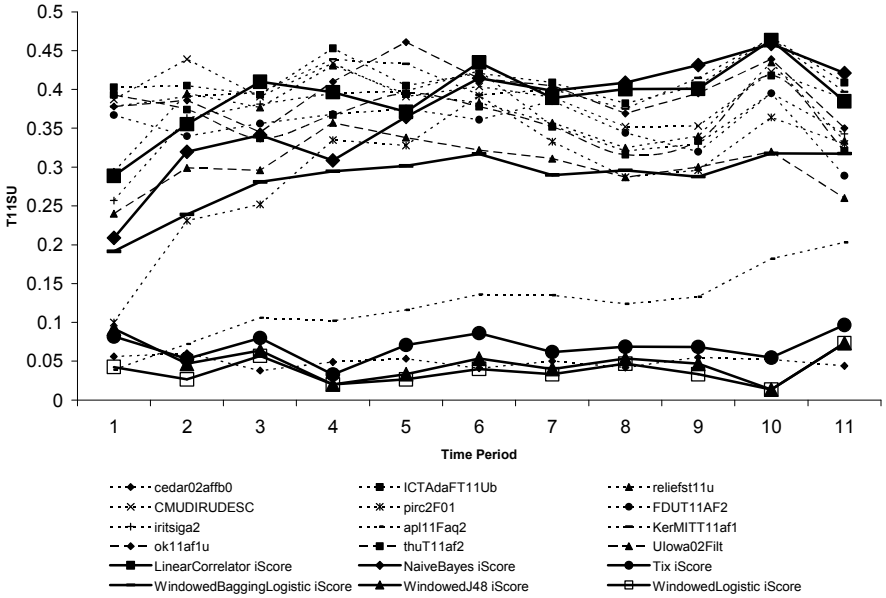


Figure 13: Performance of classifiers over time periods over the TREC articles. Each time period is a month's worth of articles.

5.2 Multiple Topic Tracking

In these set of experiments, I evaluate the efficacy of adding multiple topic tracking to the original iScore framework.

5.2.1 Case Study: Operating Parameters

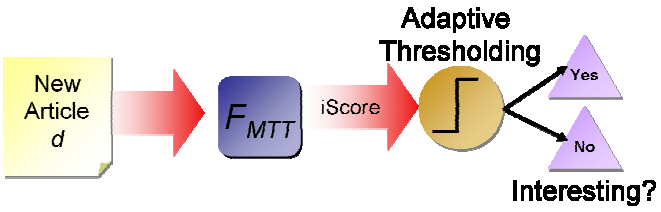


Figure 14: MTT evaluation pipeline.

To find good values for the operating parameters: $t_{cluster}$, $t_{classification}$, $t_{precision}$, γ, and M, I evaluate the resource consumption and the quality of results produced by MTT with various parameters for the "Politics Top Stories" RSS feed. For simplicity and to evaluate MTT in isolation, I use the pipeline shown in Figure 14 instead of the complete iScore pipeline. The adaptive thresholder optimizes for F-measure ($\beta = 0.5$).

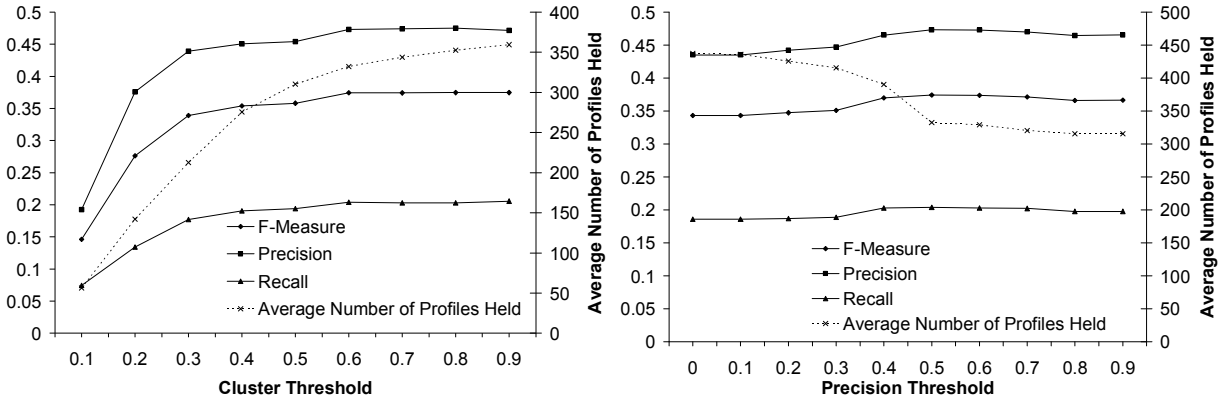
I first evaluate the effect of $t_{cluster}$ on MTT by varying $t_{cluster}$ while holding $t_{precision}$, $t_{classification}$, M, and γ at 0.5, 0.5, ∞., and 0, respectively. The results are shown in Figure 15a. As $t_{cluster}$ increases, articles are discouraged from clustering. Consequently, the average number of profiles held increases as $t_{cluster}$ increases. Low $t_{cluster}$ values cause fewer clusters to be formed, causing MTT to behave similarly as Rocchio when $t_{cluster}$ is low. The figure also shows that any $t_{cluster}$ value greater than 0.6, there is no significant increase in the quality of results while there is an increase in resource consumption. From this observation, I use 0.6 for $t_{cluster}$ for all subsequent experiments.

Next I evaluate the effect of $t_{precision}$ on MTT by varying $t_{precision}$ while holding $t_{cluster}$, $t_{classification}$, M, and γ at 0.6, 0.5, ∞., and 0, respectively. Figure 15b shows that there is little variation in performance overall when $t_{precision}$ is varied. However, there is a slight increase in performance when $t_{precision}$ increases from 0.3 to 0.5. There is also a decrease in resource consumption in the same range. Overall, fewer profile vectors are kept in memory but the profiles are more precise in identifying interesting articles when $t_{precision}$ is increased. Performance peaks when $t_{precision} = 0.5$, with performance slightly decreasing for any values beyond 0.5 with very little decrease in resource consumption. Consequently, for all later experiments, I use 0.5 for $t_{precision}$.

Next I evaluate $t_{classification}$ on MTT by varying $t_{classification}$ while holding $t_{cluster}$, $t_{precision}$, M, and γ at 0.6, 0.5, ∞., and 0, respectively. As $t_{classification}$ increases, the average number of profiles held in memory increases. Profiles are discarded when they generate too many false positives due to the minimum precision discard mechanism. Higher $t_{classification}$ values make it more difficult for the profiles to generate false positives (while generating many more false negatives), so fewer profiles are discarded. Figure 15c shows that a good value for $t_{classification}$ is 0.4. Values less than 0.4 cause profiles to generate too many false positives and are consequently discarded, so fewer profiles are kept, decreasing the number of topics that can be tracked. However, too high of a value for $t_{classification}$ will lead to too many false negative classifications, as shown in the decrease in recall in Figure 15c. Interestingly, there is also a decrease in precision for too high $t_{classification}$ values. This is most likely due to noise caused by anomalous taggings that would normally be removed when low precision profiles are discarded. It is also due to changes in user interests which are immediately addressed by removing low precision profiles as well. For all subsequent experiments, I use 0.4 for $t_{classification}$.

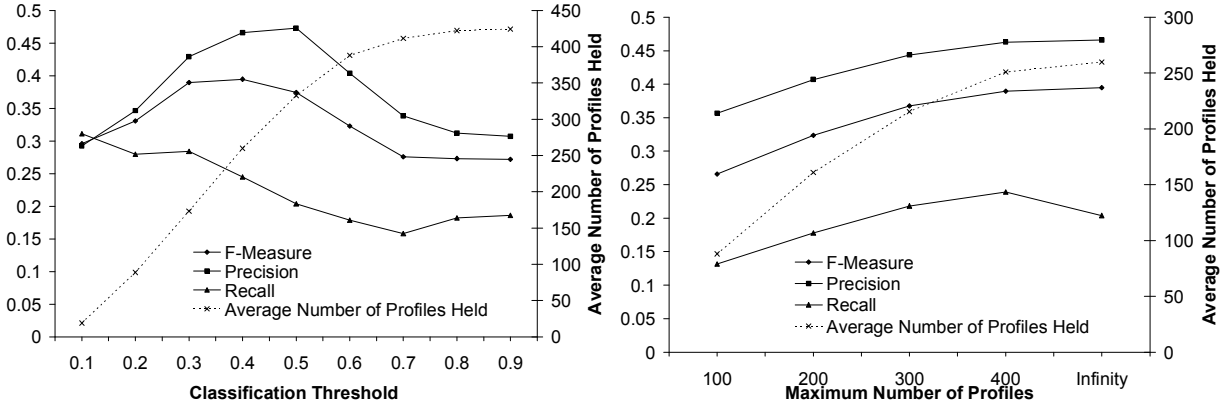
Next I evaluate the effect of M, which is the maximum number of profiles kept in memory, while holding $t_{cluster}$, $t_{precision}$, $t_{classification}$, and γ at 0.6, 0.5, 0.4, and 0, respectively. Figure 15d shows that that as the number of maximum profiles increase, the quality of results improve, with significantly higher precision. However, it is inconclusive to determine if it is better to leave the number of profiles unbounded because the number of profiles kept in memory for "Politics Top Stories" is at most 500. It is difficult to determine the tradeoff in resource consumption with performance, given the results in Figure 15d. A data set that spans a large period of time is likely needed to determine a good M value. So for all subsequent experiments, I leave the number of profiles kept in memory to be unbounded, or M=∞.

Finally, I evaluate the effect of γ, which controls how much of an effect that misclassified uninteresting articles have (compared to interesting articles), by varying γ, while holding $t_{cluster}$, $t_{precision}$, $t_{classification}$, and M at 0.6, 0.5, 0.4, and ∞., respectively. Figure 15e shows that there is very little effect caused by γ. There is only a slight increase in performance when γ =0.5. However, there is a significant change in memory consumption, as γ increases. Since the profiles are actually the centroids of clusters of interesting documents, using a large γ value, changes the natural document clusters. As more documents are



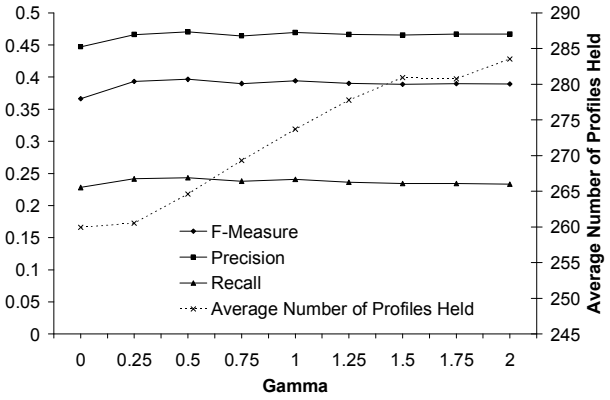
(a)

(b)



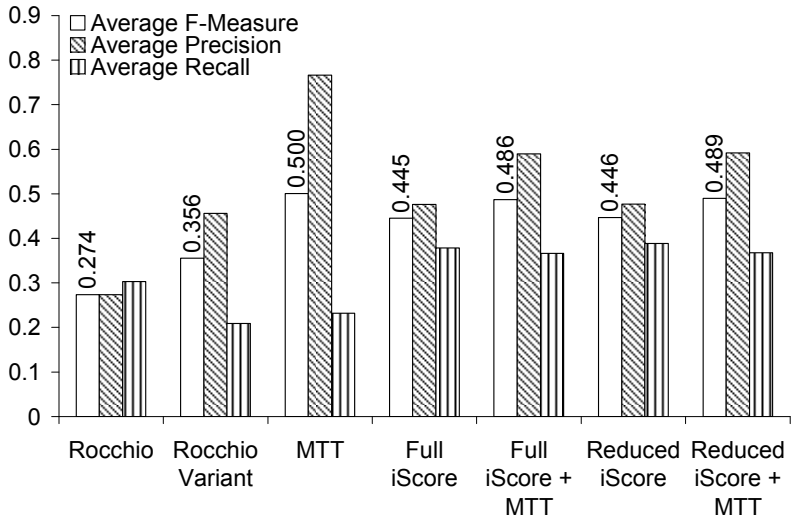
(c)

(d)

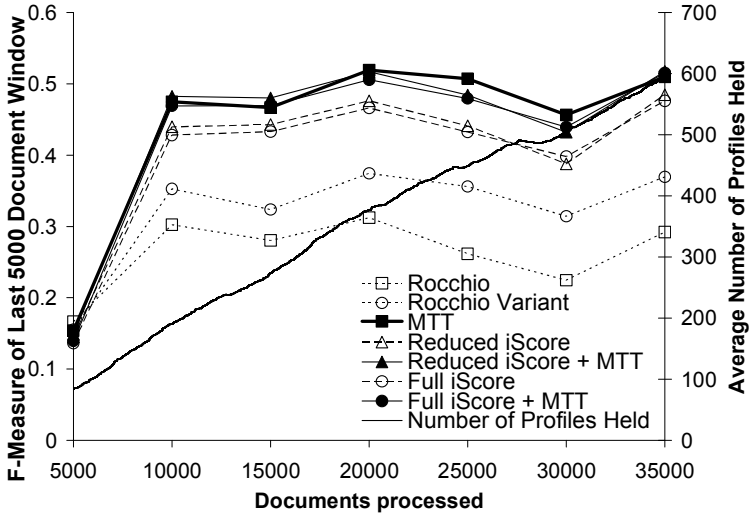


(e)

Figure 15: Operating parameters for Politics Top Stories from the Yahoo! RSS Feeds



(a)



(b)

Figure 16: Performance over time of iScore and MTT using the Yahoo! RSS Feeds. Figure 5a shows the overall performance of the classifiers after processing 10,000 documents. Figure 5b shows the performance of the classifiers over time.

processed, new clusters must be generated since the natural clusters no longer exist. However, using a small non-zero γ value can help reduce the noise in the clusters and improve the classification quality. For all subsequent experiments, I use 0.5 for γ.

A similar case study was performed for the "Technology Top Stories" RSS feed. Good values found for the feed are $t_{cluster} = 0.7$, $t_{precision} = 0.8$, $t_{classification} = 0.4$, and $M = \infty$. Using these values, varying γ had no effect. The parameter configuration seems to be user-dependent. However, for simplicity, I use the good parameters found for the "Politics Top Stories" feed for all subsequent experiments.

5.3 Overall Performance

Given the results from the case study for finding good operating parameters for the "Politics Top Stories" RSS feed, I applied MTT with the same parameters to all the other RSS feeds. In this experiment as well,

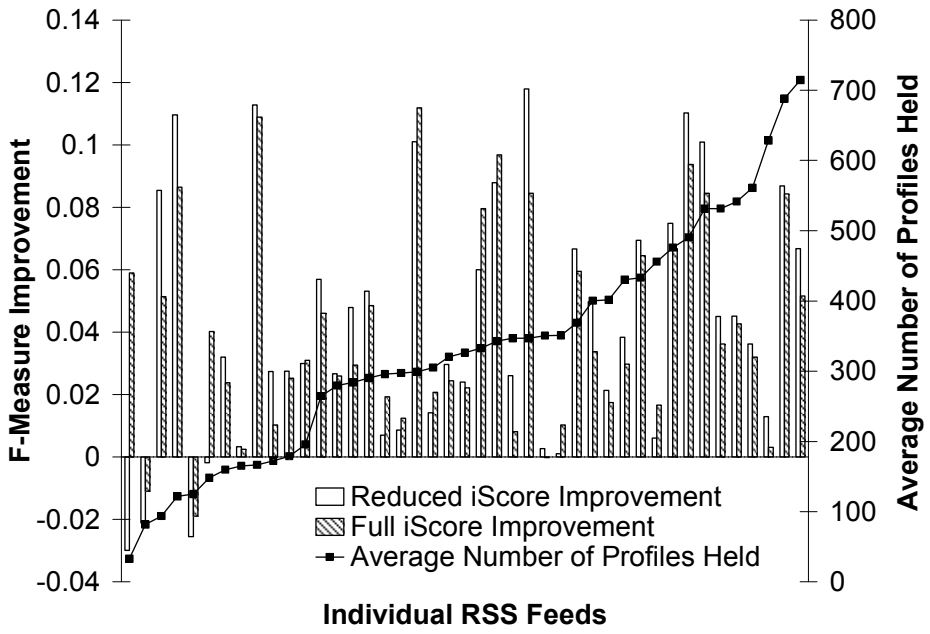


Figure 17: Overall improvement (after processing 10,000 documents) of iScore by including MTT for individual RSS feeds. Each column is an individual RSS feed.

the adaptive thresholder optimizes for F-measure. I use $t_{cluster}$=0.6, $t_{precision}$=0.5, $t_{classification}$=0.4, M=∞, and γ =0.5. The mean average results are shown in Figure 16.

In Figure 16a, I compare the overall performance of various classifiers after processing 10,000 documents, including Rocchio and the Rocchio variant from [Xu2002], which performed the best in the TREC11 adaptive filter task. The figure shows that MTT performs significantly better than the Rocchio variant, with a mean average F-measure (where β=0.5) of 0.500, performing 40% better. MTT also outperforms Rocchio by 82%. According to the t-test, MTT's improvements over Rocchio and its variant are statistically significant (p = 2.7E-11 over Rocchio, p = 1.6E-06 over the Rocchio variant).

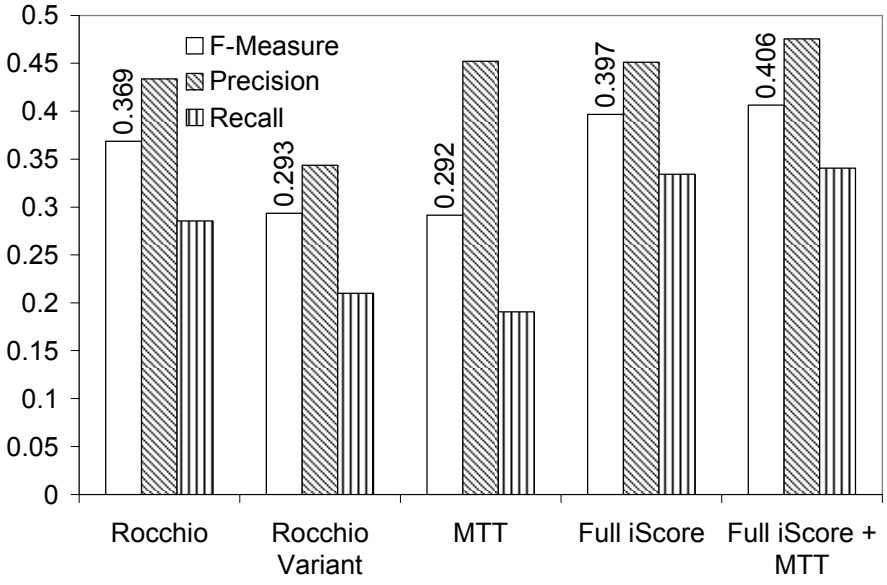
I also evaluate MTT when it is included in the complete iScore pipeline. I evaluate two iScore systems, one with the complete feature set from [Pon2007] and one with a reduced feature set with the highest correlated features to interestingness. The reduced feature set contains all the features from [Pon2007] except for freshness, new n-gram anomaly detection, and n-gram and tokenized language modelling anomaly detection. Figure 16a shows iScore with MTT has a similar F-measure performance as MTT alone, with iScore yielding greater recall and lower precision. Figure 16a also shows that the inclusion of MTT into iScore results in a statistically significant increase of 9% in F-measure (p = 0.09). Closer examination of each individual RSS feed shows positive improvement for most RSS feeds in Figure 17. RSS feeds with a lower average number of profiles held in memory due to very few interesting articles are more likely to yield negative improvement.

Figure 16b shows the performance of the classifiers over time as well as the mean average number of profiles held in memory. The F-measure dramatically increases after processing 10,000 articles. The figure also shows statistically significant improvements (p < 0.02*)* that MTT and iScore with MTT have over Rocchio, the Rocchio variant, and iScore without MTT. After processing 25,000 documents, there is a statistically significant 10% increase in performance of iScore caused by the inclusion of MTT (p = 0.02), which is consistent to what was observed in Figure 16a. Although, it seems that MTT alone performs better than iScore with MTT after processing 25,000 documents, the t-test shows that that increase is not statistically significant (p = 0.23) for both feature sets. The figure also shows that the average number of profiles held increases linearly as more documents are processed. This behavior is

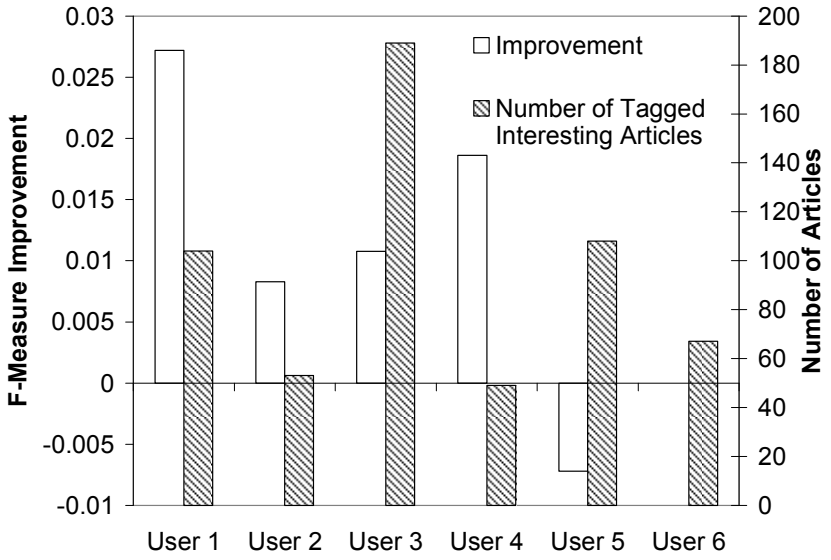
expected since new topics continually appear and the maximum number of profiles is left unbounded for these experiments so no unused interesting topics are discarded. Further study with a larger corpus is necessary to determine a good maximum, if any.

5.4 User Study

In addition to the Yahoo! RSS feed articles, I also compare the performance of MTT and iScore with Rocchio and the Rocchio variant on a collection of web pages tagged by users using a web browser plug-



(a)



(b)

Figure 18: Performance of iScore and other classifiers in the user tagging collection. Figure 18a shows the overall performance of the classifiers. Figure 18b shows the overall of iScore by including MTT for individual users along with the number of tagged articles for each user.

in. In my experiments where positive user taggings are sparse, I find that the adaptive thresholder performs better when it optimizes for T11SU instead of F-measure, so in this set of experiments, the adaptive thresholder optimizes for T11SU. Also due to the scarcity of negative user taggings compared to the size of the entire data collection, I infer additional negative user taggings for articles that the user did not read but were accessible from the referring page of an article tagged by the user. Consequently, for each user, interesting news articles are predicted from a pool of articles consisting of articles that the user actually tagged and articles that were accessible from referring pages of tagged articles. I use the same operating parameters found in my case study for the "Politics Top Stories" RSS feed.

The results of the user study are shown in Figure 18. In Figure 18a, MTT has higher precision than the Rocchio variants but has much lower recall. As a result, MTT's F-measure performance is lower than the Rocchio variants. iScore with MTT performs 10% better than Rocchio in terms of F-measure. And the inclusion of MTT improves iScore by 2%. However, the comparison of the results is difficult to determine because the t-test indicates that the comparisons are not statistically significant with $p \geq 0.25$. More users are necessary to make a definitive judgment, which is likely to be similar to the statistically significant judgments made with the Yahoo! data set. Closer inspection of each individual user shows F-measure improvement to iScore for most users when MTT is included into iScore's feature set, as shown in Figure 18b.

Although a smaller and potentially noisier data set, this limited user study shows that iScore with MTT can work well even with a limited number of user taggings. For the Yahoo! data set, each RSS feed has an average of 655 tagged articles. In contrast, this limited study, as shown in Figure 18b, has a much smaller collection of taggings with each user tagging an average of 95 interesting articles.

Although iScore with MTT performs better than traditional techniques, it performs poorer in this data set than in the Yahoo! data set, which is most likely due to the noise in the data collection caused by the inference of additional negative user taggings. Also the web pages contain peripheral information in addition to the news story, such as navigation menus and links to other web pages, which make processing the content of the news story more difficult.

5.5 TREC Filtering

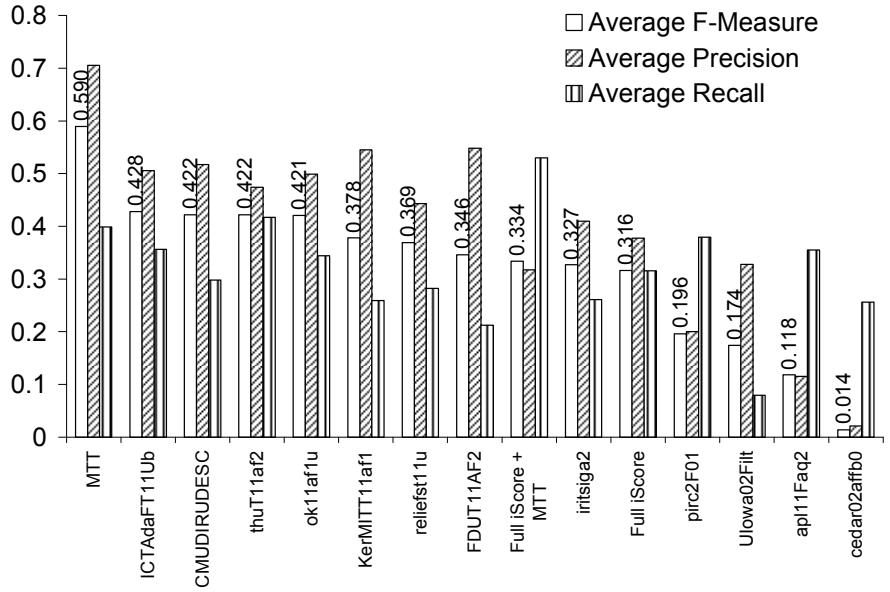
Although the TREC11 adaptive filter task is to retrieve all articles relevant to a query, regardless of its interestingness to a user, I want to see how well MTT and iScore performs against other adaptive filters from TREC11. MTT and the full iScore feature set are compared with the best filters from each participating group in TREC11 against the TREC11'S RCV1 corpus in Figure 19. As in the user tagging collection, the taggings in the TREC collection are very sparse relative to the size of the collection, so the adaptive thresholder optimizes for T11SU as well. I use the same operating parameters found in my case study for the "Politics Top Stories" RSS feed. Figure 19a shows that MTT has an F-measure 37.8% better than the best performing filter in TREC11 [Robertson2002]. MTT also yields higher precision and recall. When MTT is incorporated into iScore, F-measure improves by 9% but is not statistically significant ($p = 0.25$). According to [Pon2007], features other than topic relevancy features are not useful for identifying interesting articles to the TREC topics. The addition of irrelevant features causes the statistically significant difference in performance between MTT alone and iScore with MTT ($p < 0.01$) while only improving iScore slightly when added as an additional feature.

Figure 19b shows the performance of iScore and MTT along with the top three adaptive filters from Figure 19a. TREC only reports T11SU performance over time instead of F-measure, so T11SU is shown in Figure 19b. The figure shows that MTT performs much better over time than all the other classifiers. Like Figure 19a, Figure 19b does show statistically insignificant improvement of 2% ($p = 0.31$) for documents after time period 6.

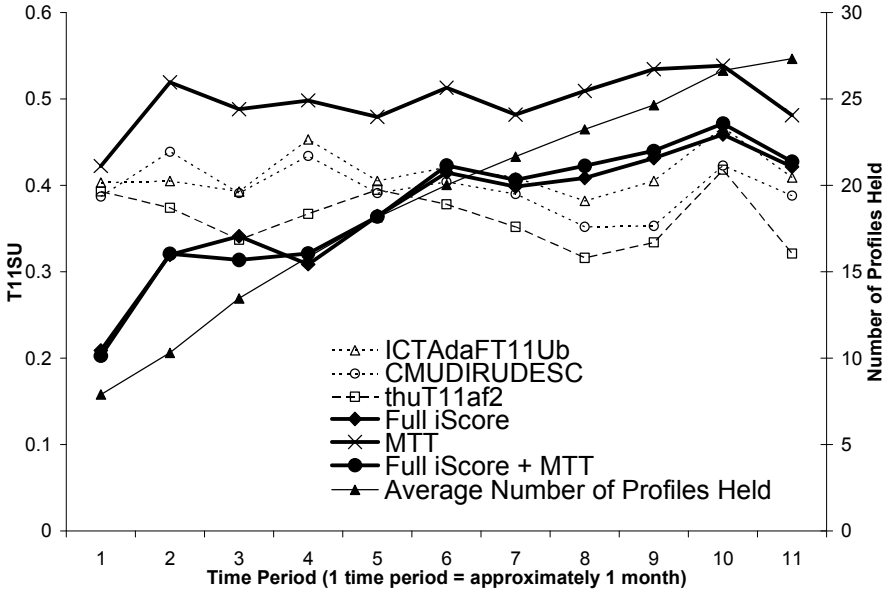
Comparing Figure 16b and Figure 19b, the resource consumption in the TREC data is much less than in the Yahoo! data. Despite the larger size of the TREC corpus, user interests in the TREC data set are much narrower than that of the Yahoo! data so fewer profile vectors are necessary to represent the entire range of user interests.

6 Possible Future Work

Although the iScore and MTT has made significant advances towards accurately identifying interesting



(a)



(b)

Figure 19: Performance of iScore and MTT in the TREC11 adaptive filter task. Figure 19a shows the overall performance of the classifiers. Figure 19b shows the performance of the classifiers over time.

articles, there are two areas of additional research, such as adaptive multi-role tracking, which is an extension of MTT, and the identification of interesting relationships and entities, leveraging work from relationship networks, that would make further advances. In addition to these areas, additional corpora will be used to thoroughly evaluate my work.

6.1 Adaptive MTT

The inclusion of MTT into iScore has improved the classification of interesting documents for most users and data sets by 10% overall. However, there is room for further study. In the presented experiments, the parameters were static for all users. In my case study, it was shown that two different users can have two different near-optimal parameters configurations. Using the parameter configuration found here as starting points and given the expected behavior of MTT for various parameters, optimal parameters tailored for specific users with specific memory constraints and quality requirements can be dynamically learned as more documents are processed so that maximum performance for each user can be achieved with MTT.

6.2 Multi-Role Users

Additionally, my work in MTT has shown that users are interested in a multitude of news topics and that not one single model can accurately capture all the interests of the user. This may also be applicable to other features. Depending on the role that a user is serving while reading the news, some features may be interpreted differently depending on the role the user is currently serving. For example, a user interested in "terrorist activity" would have a higher preference for negative polarity articles; whereas, when the same user may also be interested in "UCLA basketball" would have a higher preference positive polarity articles. Further research will look at how to construct multiple classifiers to address multiple user roles for a single user.

6.3 Identifying interesting relationships and entities

The use of entity networks or ontologies representing a user's interests maybe useful for differentiating interesting articles from uninteresting ones. [Angelova2005] presents a survey of various techniques necessary for ontology acquisition, including named entity extraction, the acquisition of concepts, and learning taxonomies. In [Singh2004], studies how binary relationships are expressed within single sentences. In each sentence, pairs of target entity types are considered as candidates and are classified with a support vector machines with a variety of features.

A preliminary study has been done to evaluate the applicability of using PageRank [Page1998] to identify important entities in news, such as persons or locations. Using this information, I can identify the most interesting articles that discuss these entities. This task entails the following three subtasks:

1. Construct a meaningful directed graph, where two connected entities are connected by a directed edge such that the source of the edge is transferring some of its "importance" to the destination of the edge. The relationship between the entities should be learned from articles or from an external knowledge-base.
2. Evaluate PageRank on the network to determine the importance of the entities in the network.
3. Using the entities' PageRank scores, evaluate any new article's "interestingness."

As a preliminary study, I studied the relationship of the PageRank scores of entities to the interestingness of their containing articles on the Yahoo! RSS feed data set for the "Top Stories Politics" feed. A graph of entities is maintained as documents are processed in chronological order of their publication date. Using Sundance/AutoSlog [Riloff2004], a shallow parser and pattern extractor NLP tool, the subjects and direct objects of sentences were extracted from each article. The subjects and direct objects extracted are added to the graph and a relationship is also added, connecting the subject to the direct object. Intuitively, the

subject is passing some of its "importance" to its direct object because it is "doing something" to its direct object. For example, in the sentence, "George Bush lunched with the Boy Scouts of America," George Bush is passing some of his importance to the Boy Scouts of America because the Boy Scouts are important enough for Bush to have lunch with them. However, it can be argued that the inverse is also true. But for this preliminary evaluation of PageRank's applicability and for simplicity, I consider only the subject to direct object relationships. Additionally, for each entity, I maintain a count of how many articles contain the entity and how many interesting articles contain the entity as well. Using these two counts, I maintain a TF-IDF statistic for each entity *e*:

$$TF - IDF_{Interesting}(e) = \frac{|\text{Interesting Articles containing } e|}{|\text{Interesting Articles Seen So Far}|} \log\left(\frac{|\text{Articles Seen So Far}|}{|\text{Articles containing } e|}\right) \quad (21)$$

Processing articles in chronological order of their publication date, the PageRank scores of all entities in the current network is computed every 7 days. However, instead of the traditional PageRank scores detailed in [Page1998], I use PageRank with priors [White2003] to give more weight to entities that I know to be more important to the user. In [White2003], PageRank is formulated as the following, where $d_{in}(v)$ is the set of entities that point to an entity v and $p(v)$ is the prior bias of v:

$$PageRank(v) = (1-\beta) * \left(\sum_{u \in d_{in}(v)} p(v \mid u) PageRank(u) \right) + \beta * p(v) \quad (22)$$

In [White2003], β=0.3 and $p(v) = \frac{1}{|R|}$ for $v \in R$, where R is the set of known important entities, $p(v)$=0 otherwise. I also use the same configuration in my experiments. Whenever the periodic PageRank scores $PageRank_t$ for time period t are computed, the set of known important entities are the current top 100 entities with the highest TF-IDF scores of the last 7 days. After all the documents are processed, the PageRank scores $PageRank_c$ of all the entities are computed on the complete final network using the top 100 entities with the highest TF-IDF scores overall. For each document, I extract the features detailed in Table I, where E is the set of all entities contained within a document, t is the last time period for which PageRank was computed and $IDF(e) = \log\left(\frac{|\text{Articles Seen So Far}|}{|\text{Articles containing } e|}\right)$.

Table I: PageRank-based features

Feature	Description
avg	$\frac{1}{\lvert E \rvert} \sum_{e \in E} PageRank_c(e)$
min	$\min_{e \in E}(PageRank_c(e))$
max	$\max_{e \in E}(PageRank_c(e))$
sum	$\sum_{e \in E} PageRank_c(e)$
stddev	$stddev_{e \in E}(PageRank_c(e))$
avgIDF	$\frac{1}{\lvert E \rvert} \sum_{e \in E} PageRank_c(e) * IDF(e)$
minIDF	$\min_{e \in E}(PageRank_c(e) * IDF(e))$
maxIDF	$\max_{e \in E}(PageRank_c(e) * IDF(e))$
sumIDF	$\sum_{e \in E} PageRank_c(e) * IDF(e)$

stddevIDF	$\underset{e \in E}{stddev}(PageRank_c(e) * IDF(e))$
avgLogIDF	$\frac{1}{\lvert E \rvert} \sum_{e \in E} \log(PageRank_c(e) * IDF(e))$
minLogIDF	$\min_{e \in E}(\log(PageRank_c(e) * IDF(e)))$
maxLogIDF	$\max_{e \in E}(\log(PageRank_c(e) * IDF(e)))$
sumLogIDF	$\sum_{e \in E} \log(PageRank_c(e) * IDF(e))$
stddevLogIDF	$\underset{e \in E}{stddev}(\log(PageRank_c(e) * IDF(e)))$
avgLog	$\frac{1}{\lvert E \rvert} \sum_{e \in E} \log(PageRank_c(e))$
minLog	$\min_{e \in E}(\log(PageRank_c(e)))$
maxLog	$\max_{e \in E}(\log(PageRank_c(e)))$
sumLog	$\sum_{e \in E} \log(PageRank_c(e))$
stddevLog	$\underset{e \in E}{stddev}(\log(PageRank_c(e)))$
minDelta	$\min_{e \in E}(PageRank_t(e) - PageRank_{t-1}(e))$
maxDelta	$\max_{e \in E}(PageRank_t(e) - PageRank_{t-1}(e))$
sumDelta	$\sum_{e \in E}(PageRank_t(e) - PageRank_{t-1}(e))$
avgDelta	$\frac{1}{\lvert E \rvert} \sum_{e \in E}(PageRank_t(e) - PageRank_{t-1}(e))$
stddevDelta	$\underset{e \in E}{stddev}(PageRank_t(e) - PageRank_{t-1}(e))$
count	|E|

To evaluate how well PageRank could be used to infer the "interestingness" of articles, I evaluate the features extracted using WEKA's Knowledge Flow tool with the flow shown in Figure 20. I ran a 10-fold cross-validation on the data set for the interesting articles defined by the "Top Stories Politics" feed and used AdaBoosting [Freund1996] on a C4.5 decision tree generated by the J48 algorithm.

The precision-recall curve for the classifier using the extracted features to predict the interestingness of news articles is shown in Figure 21. The graph shows that it is possible attain 50% precision, with 10% recall. Although, the performance of this classifier is low compared to methods such as MTT and iScore as a whole, it may be useful as an additional feature into iScore if it provides another dimension not already incorporated into iScore. Using two feature selection criterions: information gain and chi-squared rankings, I determined the usefulness of the PageRank features. For the information gain rankings, minDelta, maxDelta, sumDelta, maxIDF, and maxLogIDF were the top 5 most useful features. For the chi-squared rankings, minDelta, maxDelta, sumDelta, avgDelta, and stddevDelta were the top 5 most discriminating features. These two rankings indicate that the changes in the importance of entities in news are more useful than static PageRank scores for determining which articles are interesting, which is a dimension not already incorporated into iScore

Although, PageRank features show some promise for helping determine the "interestingness" of articles and would add another dimension to iScore, PageRank is costly to compute frequently, especially if the network is continually growing as more documents are processed. Additional research will need to focus on less costly estimations of PageRank [Cho2007] so it may be adapted into a streaming setting. More research will also need to be done to identify appropriate time periods to evaluate PageRank scores and the appropriate size for the prior nodes set.

Using this relationship network, it may also be possible to adapt work from [Rattigan2005] to identify anomalous links. In [Rattigan2005], the authors make a case for the usefulness of the identification of anomalous links among authors in determining the interestingness of conference articles. Anomalous links in my network may indicate previously unseen relationships among persons or locations. For example, before September 11, airplanes and the World Trade Center were previously unconnected, but was highly interesting news when the terrorist attacks first occurred.

6.4 Large Article Collections Studies

More news articles from the Yahoo! RSS feeds are being collected, so that iScore can be evaluated over a larger corpus. In my case study, due to the size of the Yahoo! RSS feed used to evaluate MTT and the number of relevant articles in the TREC11 adaptive task, the resource usage behavior of MTT when the maximum number of profiles is varied could not be accurately determined. A larger Yahoo! RSS feed collection spanning a large time period would help determine a good value, if any. More user taggings of articles by volunteers are being collected to improve the quality of the user tagging data set as well. As of May 1, 2007, there have been 125,871 news articles collected from the Yahoo RSS feeds.

Although the Yahoo RSS feed article collection provides a good data set to evaluate iScore upon, it models users as a collective community. For example, the "Top Stories Politics" feed serves as a proxy for a user who is interested in politics. However, the user is modeled after a community instead of as an individual interested in politics. Consequently, in addition to the Yahoo RSS feeds, I are also collecting news articles and taggings from volunteer users using a Firefox plugin. As of April 12, 2007, there have been 16,071 news articles for eight users having tagged more than 40 articles each. However, due to the difficulty in recruiting to consistently read and tag articles, I have looked at an alternative method for collected via the web service, Digg, to complement my user tagging collection. As of April 12, 2007, there have been 13,856 pages collected "dugg" [Digg07] by the top 100 active Digg users [Finke2007]. Although these two data sets would accurately reflect individual users, the data set is much smaller and much noisier due to the heterogeneity of the type of pages being collected. Table II shows a summary of the data sets that will be used to evaluate iScore along with their advantages and disadvantages. With this much larger document collection, I am hoping to rerun similar experiments so that I can reach much more statistically significant conclusions.

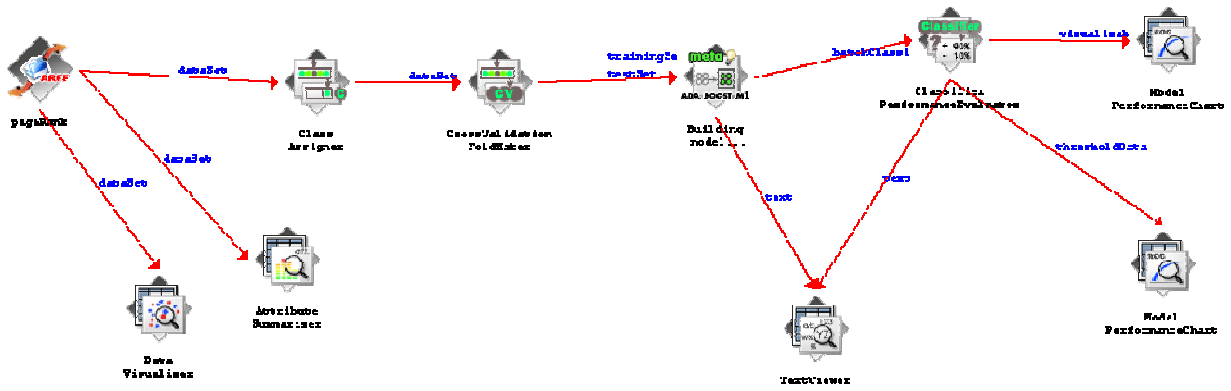


Figure 20: Knowledge Flow in WEKA .

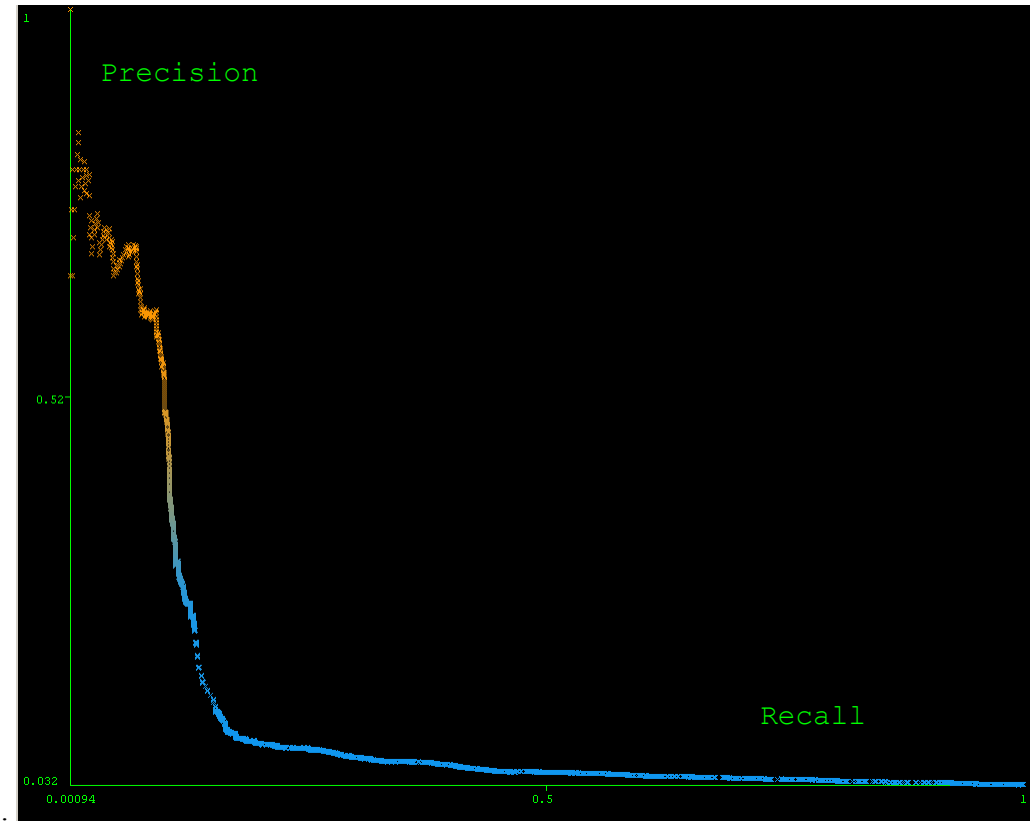


Figure 21: Precision-Recall curve for PageRank featured classifier.

Table II: News collections

Data Set	Description	Size	Advantage/Disadvantage
TREC11 Adaptive Filter	Reuters RCV1 Corpus	723,431 articles	Advantage: Very large corpus Disadvantage: Can only evaluate iScore on topic relevancy only
Yahoo RSS Feeds	News articles collected for feeds such as Top Stories Politics or Most Emailed Technology	125,871 articles (4/12/07)	Advantage: Large corpus, easily collected, and very clean data. Disadvantage: Can only evaluate iScore for users modeled as a collective community
Digg Articles	Articles "dugg" by the top 100 Digg users	13,856 articles (4/12/07)	Advantage: Easily collected, evaluates iScore on individual users Disadvantage: Small corpus, and very noisy

			data
Volunteer User Tagged Articles	Articles tagged by volunteers along with the articles contained within the referring page of the tagged article.	16,071 articles (4/12/07)	Advantage: Evaluates iScore on individual users Disadvantage: Difficult to collect, small corpus, and very noisy data

6.5 Timeline

Table III details the timeline for future work. Expected completion of my dissertation is Summer 2008.

Table III: Research Timeline

Item	Start Date	Expected Completion Date
Evaluate current iScore implementation on large collections	Beginning of Summer 2007	End of Summer 2007
Relationship/Entity	After oral examination	End of Fall 2007
Adaptive MTT or Multi-role users	Beginning of Winter 2007	End of Spring 2008
Final Defense		End of Summer 2008

7 Conclusion

Having implemented a prototype of the iScore framework, I show 21% to 24% improvement over traditional IR techniques for identifying interesting articles, indicating that filtering based on only topic relevancy is insufficient for identifying interesting articles. Extracting a variety of features, ranging from topic relevancy to source reputation, I show that no single feature can characterize the interestingness of an article for a user. It is the combination of multiple features that yields higher quality results. For each user, these features have different degrees of usefulness for predicting interestingness. Within the iScore framework, I evaluate several classifiers for combining these features to find an overall interestingness score. Through user-feedback, the classifiers find features that are useful for predicting interestingness for the user. After studying the correlation distributions of the Yahoo RSS Feeds and the TREC adaptive filter documents, I find that current evaluation corpora, such as TREC, do not capture all aspects of personalized news filtering systems necessary for system evaluation.

Within the iScore framework, I adapt multiple topic tracking typically used in TDT, improving performance on the Yahoo! RSS Feeds by 10% compared to iScore without MTT. Instead of identifying all new topics and tracking all articles for those topics as in TDT, I focus on the specific users interests, which are under continuous evolution. Focusing on only evolving user interests instead of all topics allows for more efficient resource utilization. I show that the use of multiple profile vectors yield significantly better results than traditional methods, such as the Rocchio algorithm, for identifying interesting articles. Additionally, the addition of tracking multiple topics as a new feature in iScore, improves iScore classification performance. For a specific user as a case study, I identify the operating parameters for my algorithm for their resource usage and classification performance. I show that multiple topic tracking yields 37.8% better results than the best results from the last TREC adaptive filtering run.

Future work will focus on multiple areas. Having seen evidence that a single user profile model may not be able to capture all the interests of a user, I can study how to construct multiple classifiers to address multiple user roles for a single user. Another area of work can look on how user-specific operating

parameters for MTT can be learned dynamically. Future work will focus on one of these two areas. Preliminary research will help me determine which area has the best payoff. Also having seen some promising results in applying PageRank to the interesting news article problem, future work will focus on identifying interesting entities and relationships extracted from news articles, in hopes that it may provide another dimension to what makes an article interesting. More extensive performance study will be done using a larger Yahoo! RSS Feed news article collection, a large news article collection from Digg, and volunteer user-tagged articles.

8 References

[Alias-I2006] Alias-I, "LingPipe," September 2006. [Online]. Available: http://www.alias-i.com/lingpipe/index.html

[Allan1998] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. New York, NY, USA: ACM Press, 1998, pp. 37-45.

[Allan2002] J. Allan, "Detection as multi-topic tracking," *Inf. Retr.*, vol. 5, no. 2-3, pp. 139-157, 2002.

[Al-Mubaid2006] H. Al-Mubaid and S.A. Umair, "A New Text Categorization Technique Using Distributional Clustering and Learning Logic," *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1156-1165, 2006.

[Angelova2005] G. Angelova, "Language Technologies Meet Ontology Acquisition," *Lecture Notes in Computer Science*, pp. 367-380, Springer Berlin/Heidelberg, 2005.

[Angelova2006] R. Angelova and G. Weikum, "Graph-based text classification: learn from your neighbors," in *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 485{492, New York, NY,2006.

[Angiulli2005] F. Angiulli, S. Basta, and C. Pizzuti, "Detection and prediction of distance-based outliers," in *Proc. of the 2005 ACM symp on Applied computing*, 2005, pp. 537.

[Billsus2000] D. Billsus, M. J. Pazzani, and J. Chen, "A learning agent for wireless news access," in *Proc. of the 5th Intl. Conf. on Intelligent user interfaces*, 2000, pp. 33–36.

[Blei2003] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, 3:993-1022, 2003.

[Blum1997] A. L. Blum and P. Langley. "Selection of relevant features and examples in machine learning," *Artif. Intell.*, 97(1-2):245{271, 1997.

[Breiman1996] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123, 1996.

[Brouard2002] C. Brouard, "Clips at TREC-11: Experiments in filtering," in *TREC11*, 2002.

[Carreira2004] R. Carreira, J. M. Crato, D. Goncalves, and J. A. Jorge, "Evaluating adaptive user profiles for news classification," in *Proc. of the 9th Intl. Conf. on intelligent user interface*, 2004, pp. 206–212.

[Cessie1992] S. le Cessie and J. van Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, vol. 41, no. 1, pp. 191–201, 1992.

[Chase2006] P. J. Chase and S. Argamon, "Stylistic text segmentation," in Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval, 2006, pp. 633–634.

[Chaudhary2002] A. Chaudhary, A. S. Szalay, and A. W. Moore, "Very fast outlier detection in large multidimensional data sets," in *DMKD*, 2002.

[Chen1996] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proc. of the 34th annual meeting on Association for Computational Linguistics*. Morristown, NJ: Association for Computational Linguistics, 1996, pp. 310–318.

[Chesley2006] P. Chesley, B. Vincent, L. Xu, and R. K. Srihari, "Using verbs and adjectives to automatically classify blog sentiment," in *Proc. AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2006.

[Cho2007] J. Cho and U. Schonfeld, "RankMass Crawler: A Crawler with High PageRank Coverage Guarantee," *VLDB '07*, Vienna Austria, September 23-28, 2007.

[Corso2005] G. M. D. Corso, A. Gulli, and F. Romani, "Ranking a stream of news," in *Proc. of the 14th Intl. Conf. on World Wide Web*, 2005, pp. 97.

[Damianos2003] L. Damianos, S. Wohlever, R. Kozierok, and J. Ponte, "Mitap: A case study of integrated knowledge discovery tools," *HICS*, vol. 03, p. 69c, 2003.

[Denning2006] P. J. Denning, "Infoglut," *Comm. ACM*, vol. 49, no. 7, pp. 15–19, 2006.

[Diaz2006] F. Diaz and D. Metzler, "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*, pp. 154-161, New York, NY, 2006.

[Digg2007] Digg, "Digg," April 2007,[Online] http://www.digg.com

[Dragomir] Dragomir, R. Weiguo, and F. Zhu, "Webinessence: A personalized web-based multidocument summarization and recommendation system," [Online]. Available: citeseer.ist.psu.edu/dragomir01webinessence.html

[Eskin2000] E. Eskin, "Detecting errors within a corpus using anomaly detection," in *Proc. of the 1st Conf. on North American chapter of the Assc. for Computational Linguistics*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 2000, pp. 148–153.

[Esposito2004] R. Esposito and L. Saitta. "A Monte Carlo analysis of ensemble classification," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, p. 34, New York, NY, 2004.

[Fink2007] C. Finke, "Digg's Top 100 Users," April 2007, [Online] http://www.efinke.com/digg/topusers.html

[Fogg2003] B. J. Fogg. "Prominence-interpretation theory: explaining how people assess credibility online," in CHI '03: CHI '03 extended abstracts on Human factors in computing systems, pp. 722-723, New York, NY,, 2003.

[Forman2004] G. Forman, "A pitfall and solution in multi-class feature selection for text classification," in *Proc. 21st Intl. Conf. on Machine Learning*, 2004, p. 38.

[Forman2006] G. Forman, "Tackling concept drift by temporal inductive transfer," in Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in

information retrieval, 2006, pp. 252–259.

[Franz2001] M. Franz, T. Ward, J. S. McCarley, and W.-J. Zhu, "Unsupervised and supervised clustering for topic tracking," in SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. New York, NY, USA: ACM Press, 2001, pp. 310-317.

[Freund1996] Y. Freund and R.E. Schapire, "Experiments with a new boosting algorithm," *International Conference on Machine Learning*, 1996, pp. 148-156.

[Genkin2004] A. Genkin, D.D. Lewis, and D. Madigan, "Large-Scale Bayesian Logistic Regression for Text Categorization," *Journal of Machine Learning Research*, 2004.

[Griffiths2004] T.L. Griffiths and M. Steyvers, "Finding scientific topics," *Proc Natl Acad Sci USA,* 101 Suppl 1, pp 5228-523, April 2004.

[Guyon2003] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[Guyon2003] I. Guyon and Andre Elisseeff. "An introduction to variable and feature selection," J. Mach. Learn. Res., 3:1157-1182, 2003.

[Hayes-Roth2006] F. Hayes-Roth, "Two theories of process design for information superiority: Smart pull vs. smart push," in *Proc. of Command and Control Research and Technology Symp.: The State of the Art and the State of the Practice*, San Diego, CA, 2006.

[Henzinger2006] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006, pp. 284–291.

[IBM2006] IBM, "Unstructured information management architecture SDK," September 2006. [Online]. Available: http://www.alphaworks.ibm.com/tech/uima

[Joachims1996] T. Joachims, "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization," *Technical Report CMU-CS-96-118*, Carnegie Mellon University, 1996.

[John1995] G. H. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proc. of the 11th Conf. on Uncertainty in Artificial Intelligence*, 1995, pp. 338–345.

[Kohavi1997] R. Kohavi and G.H. John. \Wrappers for feature subset selection," *Artif. Intell.*, 97(1-2):273-324, 1997.

[Koppel2006] M. Koppel, J. Schler, S. Argamon, and E. Messeri, "Authorship attribution with thousands of candidate authors," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006, pp. 659–660.

[Lai2003] H.-J. Lai, T.-P. Liang, and Y. C. Ku, "Customized internet news services based on customer profiles," in *Proc. of the 5th Intl. Conf. on Electronic commerce*, 2003, pp. 225–229.

[Landwehr2005] N. Landwehr, M. Hall, and E. Frank, "Logistic Model Trees," *Mach. Learn.*, 59(1-2):161-205, 2005.

[Lazarevic2005] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceeding of the 11th ACM SIGKDD Intl. Conf. on Knowledge discovery in data*

mining, 2005, pp. 157–166.

[Li2006] J. Li, R. Zheng, and H. Chen, "From fingerprint to writeprint," *Comm. ACM*, vol. 49, no. 4, pp. 76–82, 2006.

[Lia2002] Y. Liao and V. R. Vemuri, "Using Text Categorization Techniques for Intrusion Detection," in *Proceedings of the 11th USENIX Security Symposium*, pp. 51-59, Berkeley, CA, 2002.

[Linang2004] J. Liang. "SVM multi-classifier and Web document classification," in *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 3, pp. 1347-1351, 2004.

[Littlestone1998] N. Littlestone, "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm," *Mach. Learn.*, 2(4):285-318, 1988.

[Liu2005] H. Liu, et al. "Evolving feature selection," Intelligent Systems, IEEE, 20(6):64-76, November - December 2005.

[Liu2005a] H. Liu and L. Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering," *IEEE Transactions on Knowledge and Data Engineering,* 17(4):491-502, 2005.

[Ma2002] L. Ma, Q. Chen, S. Ma, M. Zhang, and L. Cai, "Incremental learning for profile training in adaptive document filtering," in *TREC11*, 2002.

[Macskassy2001] S. A. Macskassy and F. Provost, "Intelligent information triage," in Proc. of the 24th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval, 2001, pp. 318–326.

[Makkonen2004] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi, "Simple semantics in topic detection and tracking," *Inf. Retr.*, vol. 7, no. 3-4, pp. 347-368, 2004.

[Melville2004] P. Melville and R. J. Mooney. "Diverse ensembles for active learning," in *ICML '04: Proceedings of the twenty-first international conference on Machine* learning, p. 74, New York, NY, 2004.

[Mishne2005] G. Mishne, "Experiments with mood classification in blog posts," in Style2005 - 1st Workshop on Stylistic Analysis of Text For Information Access at SIGIR, 2005.

[Nagura2006] R. Nagura, Y. Seki, N. Kando, and M. Aono. "A method of rating the credibility of news documents on the web," *in SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 683-684, New York, NY, 2006.

[Newman2006] D. Newman, C. Chemudugunta, P. Smyth, and M. Steyvers. "Analyzing entities and topics in news articles using statistical topic models," *in IEEE International Conference on Intelligence and Security Informatics*, 2006.

[NIST2004] NIST, "The topic detection and tracking 2004 (tdt-2004) evaluation project," December 2004. [Online]. Available:http://www.nist.gov/speech/tests/tdt/tdt2004/index.htm

[OpenNLP2006] OpenNLP, "OpenNLP," September 2006. [Online]. Available: http://opennlp.sourceforge.net/

[Page1998] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," *Stanford Digital Library Technologies*

Project, 1998.

[Peng2003] F. Peng, D. Schuurmans, and S. Wang, "Language and task independent text categorization with simple language models," in Proc. of the 2003 Conf. of the North American Chapter of the Association for Computational Linguistics on Human Language Technology. Morristown, NJ: Association for Computational Linguistics, 2003, pp. 110–117.

[Pilla2005] R.S. Pilla, C. Loader, and C.C. Taylor, "New Technique for Finding Needles in Haystacks: Geometric Approach to Distinguishing between a New Source and Random Fluctuations," *Physical Review Letters*, 95(23):230202, 2005.

[Pon2005] R.K. Pon and A.F. Cárdenas. "Data quality inference," in *IQIS '05: Proceedings of the 2nd international workshop on Information quality in information system*s, pp. 105-111, New York, NY, 2005. ACM Press.

[Pon2007] R. K. Pon, A. F. Cárdenas, D. Buttler, and T. Critchlow, "iScore: Measuring the interestingness of articles in a limited user environment," in *IEEE Symposium on Computational Intelligence and Data Mining 2007*, Honolulu, HI, April 2007.

[Pon2007a] R.K. Pon, A.F. Cárdenas, D. Buttler, and T. Critchlow, "Tracking Multiple Topics for Finding Interesting Articles," under review for *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, August 12-15, 2007.

[Porter1980] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[Quinlan1993] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[Rattigan2005] M. Rattigan and D. Jensen, "The case for anomalous link detection," in 4th Multi-Relational Data Mining Workshop, 11th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining, 2005.

[Riloff2004] E. Riloff and W. Phillips, "An introduction to the Sundance and AutoSlog Systems," *University of Utah School of Computing*, UUCS-04-015, 2004.

[Roberts2006] D.L. Roberts and T. Eilassi-Rad, "A position paper: value of information for evidence detection," in *AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection*, Arlington, VA, 2006.

[Robertson2002] S. Robertson and I. Soboroff, "The TREC 2002 filtering track report," in *TREC11*, 2002.

[Robertson2002a] S. Robertson, S. Walker, H. Zaragoza, and R. Herbrich, "Microsoft Cambridge at TREC 2002: Filtering track," in *TREC11*, 2002.

[Rocchio1971] J. Rocchio, *Relevance Feedback in Information Retrieval*. Prentice-Hall, 1971, ch. 14, pp. 313–323.

[Schapire1998] R. E. Schapire, Y. Singer, and A. Singhal, "Boosting and Rocchio applied to text filtering," in SIGIR '98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. New York, NY, USA: ACM Press, 1998, pp. 215-223.

[Singh2004] N. Singh, "The Use of Syntactic Structure in Relationship Structure," Massachusetts Institute of Technology Masters Thesis, 2004.

[Singhal1997] A. Singhal, M. Mitra, and C. Buckley, "Learning routing queries in a query zone," in Proceedings of the Twentieth Annual Internal ACM SIGIR Conference on Research and Development in Information Retrieval, July 1997, pp. 25-32.

[Smucker2006] M.D. Smucker and J. Allan, "Find-similar: similarity browsing as a search tool," in S*IGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 461-468, New York, NY, 2006.

[Steyvers2006] M. Steyvers. "Latent Semantic Analysis: A Road to Meaning, *Probabilistic Topic Models*. Laurence Erlbaum, 2006.

[Turinksy2004] A.L. Turinsky and R.L. Grossman. "A greedy algorithm for selecting models in ensembles," in *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 547-550, November 2004.

[Tweedie1998] F.J. Tweedie and R.H. Baayen, "How variable may a constant be? Measures of lexical richness in perspective," *Computers and the Humanities*, 32:323-352, 1998.

[Utgoff1997] P.E. Utgoff, N.C. Berkman and J.A. Clouse, "Decision Tree Induction Based on Efficient Tree Restructuring," *Machine Learning* 29(1), 1997.

[Wang2006] J. Wang, A. P. de Vries, and M. J. T. Reinders, "Unifying user-based and item-based collaborative filtering approaches by similarity fusion," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006, pp. 501–508.

[White2003] S. White and P. Smyth, "Algorithms for estimating relative importance in networks," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 266-275.

[Wiebe2000] J. Wiebe, "Learning subjective adjectives from corpora," in Proc. of the 17th Ntl. Conf. on Artificial Intelligence and Twelfth Conf. on Innovative Applications of Artificial Intelligence, 2000, pp. 735–740.

[Wiebe2002] J. Wiebe, "Instructions for annotating opinions in newspaper articles," Department of Computer Science, University of Pittsburgh," TR-02-101, 2002.

[Wiebe2004] J. Wiebe, T. Wilson, R. Bruce, M. Bell, and M. Martin, "Learning subjective language," *Comput. Linguist.*, vol. 30, no. 3, pp. 277–308, 2004.

[Witten2004] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco: Morgan Kaufmann, 2004.

[Wong1996] J.W.T. Wong, W.K. Kan, and G. Young. "ACTION: automatic classification for full-text documents," *SIGIR Forum*, 30(1):26{41, 1996.

[Wu2002] L. Wu, X. Huang, J. Niu, Y. Xia, Z. Feng, and Y. Zhou, "FDU at TREC2002: Filtering, Q&A, web and video tasks," in *TREC11*, 2002.

[Xu2002] H. Xu, et al., "TREC-11 experiments at CAS-ICT Filtering and web," in *TREC11*, 2002.

[Yahoo2007] Yahoo News, "Yahoo News – RSS," April 2007, [Online] http://news.yahoo.com/rss

[Yan2006] R. Yan and A. G. Hauptmann, "Probabilistic latent query analysis for combining multiple retrieval sources," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on*

Research and development in information retrieval, 2006, pp. 324–331.

[Yang1997] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. of the 14th Intl. Conf. on Machine Learning*. San Francisco, CA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[Yang1997] Y. Yang and J.O. Pedersen. "A Comparative Study on Feature Selection in Text Categorization," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 412-420, San Francisco, CA, 1997.

[Yu2003] H. Yu, C. Zhai, and J. Han, "Text classification from positive and unlabeled documents," in *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pp. 232-239, New York, NY, 2003.