Information Processing and Management 47 (2011) 97–116

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

# Measuring the interestingness of articles in a limited user environment

R.K. Pon [a,*], A.F. Cárdenas [a], D.J. Buttler [b], T.J. Critchlow [c]

[a] University of California, Los Angeles, 420 Westwood Plaza, Los Angeles, CA 90095, United States
[b] Lawrence Livermore National Laboratory, 7000 East Ave., Livermore, CA 94550, United States
[c] Pacific Northwest National Laboratory, PO Box 999, Richland, WA 99352, United States

ARTICLE INFO

*Article history:*
Received 15 July 2009
Received in revised form 4 March 2010
Accepted 5 March 2010
Available online 24 March 2010

*Keywords:*
News filtering
Personalization
News recommendation

ABSTRACT

Search engines, such as Google, assign scores to news articles based on their relevance 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. Relevance 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.

A general framework, called iScore, is presented for defining and measuring the "interestingness" of articles, incorporating user-feedback. iScore addresses the various aspects of what makes an article interesting, such as topic relevance, uniqueness, freshness, source reputation, and writing style. It employs various methods, such as multiple topic tracking, online parameter selection, language models, clustering, sentiment analysis, and phrase extraction to measure these features. Due to varying reasons that users hold about why an article is interesting, an online feature selection method in naïve Bayes is also used to improve recommendation results. iScore can outperform traditional IR techniques by as much as 50.7%. iScore and its components are evaluated in the news recommendation task using three datasets from Yahoo! News, actual users, and Digg.



## 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 to them. 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" (Denning, 2006), calling for a valued information at the right time (VIRT) (Hayes-Roth, 2006) strategy for information handling.

The most obvious approach for a VIRT strategy is to learn keywords of interest for a user (Billsus, Pazzani, & Chen, 2000; Carreira, Crato, Goncalves, & Jorge, 2004; Lai, Liang, & Ku, 2003). 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 135,000 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.

* Corresponding author. Tel.: +1 650 693 0971.
*E-mail addresses:* rpon@cs.ucla.edu (R.K. Pon), cardenas@cs.ucla.edu (A.F. Cárdenas), buttler1@llnl.gov (D.J. Buttler), terence.critchlow@pnl.gov (T.J. Critchlow).

0306-4573/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2010.03.001

Although it has been shown that collaborative filtering can aid in personalized recommendation systems (Wang, de Vries, & Reinders, 2006), 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, a different approach is taken by directly addressing what makes an article interesting, going beyond simple topic relevance.

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. Saracevic (1996, 2007) expands the traditional definition of relevance to include five classes of relevance: system, topical, cognitive, situational, and motivational. "Interestingness" is related to the latter three classes. A news article's informativeness, novelty, and quality contribute to its cognitive relevance. A news article's usefulness in decision making and reduction of uncertainty contribute to its situational relevance. How an article relates to the user's intents and goals contribute to its motivational relevance. Clearly, "interestingness" is much more than the traditional understanding of relevance (i.e., system and topic relevance).

To simply train a separate model per user using standard topic relevance techniques is insufficient, as additional features, such as source reputation, freshness, and writing style, are also important in determining what makes an article interesting. Furthermore, not all users are the same so the salient article features would be different among different users, necessitating a need for some online feature selection method that is user-based. There has been much work in news recommendation systems, but none have yet addressed the question of what makes an article interesting and how to build such a system.

To address the news recommendation problem, an effective online news filtering agent is needed. More specifically, the news recommendation problem addressed here follows a similar evaluation model as in the TREC11 adaptive filter task (Robertson & Soboroff, 2002). An online news agent evaluates the interestingness/relevance of articles, one at a time, in publication order. The agent is given no prior training data so it must learn as documents are streamed to it. Only when the agent makes a determination regarding the interestingness/relevance of the article is the actual interestingness/relevance revealed to the agent. The agent then is allowed to update itself with this new knowledge in preparation for the evaluation of the next article.

This paper presents a comprehensive summary of work in news recommendation for a limited user environment, from an in-depth discussion of features that would make an article interesting to how features are selected for online classification. iScore, first published by Pon, Cárdenas, Buttler, and Critchlow (2007a), the presented online news filtering agent, makes the following contributions to the problem:

1. Filtering based on only topic relevance is shown to be insufficient for identifying interesting articles.
2. Further discussion and tractable implementations of the relevance classes discussed by Saracevic (1996, 2007).
3. A variety of features are extracted, ranging from topic relevance to source reputation. No single feature can characterize the interestingness of an article for a user. It is the combination of multiple features that can yield 50% higher quality results. For each user, these features have different degrees of usefulness for predicting interestingness.
4. Several classifiers are evaluated for combining these features to find an overall interestingness score.
5. Through user-feedback, a coupled online feature selection technique/classifier finds features that are useful for predicting interestingness for the user.

In the next section, areas of related works are discussed and contrasted to the work done in iScore. The following sections discuss the overall iScore architecture, features for classification, and online classification and feature selection in iScore. The final section discusses the experimental results, evaluating iScore with traditional information retrieval (IR) techniques.

## 2. Related words

In this section, related works are discussed and contrasted with iScore. iScore's primary contributions, in relation to the following related works, are threefold:

1. iScore adapts existing online solutions, such as Rocchio and language models, that are well suited to some of the sub-problems of "interestingness." And instead of applying these solutions to their intended applications, iScore applies them to the broader problem of identifying interesting articles.
2. iScore has new and unique online solutions to sub-problems of "interestingness" that existing solutions are not tractable in an online environment, such as multiple topic tracking, novelty detection, and online feature selection. And these solutions are also applied to the general problem of finding interesting articles.
3. iScore is an ensemble system that combines multiple techniques that address different aspects of "interestingness."

### 2.1. 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. Corso, Gulli, and Romani (2005) rank 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 iScore, news articles are ranked based on various properties in an online method, but instead of ranking articles using mutual reinforcement and article freshness, a different variety of features is studied. Additionally, when training the classifiers in iScore, the likeliness that the most recent news articles are more important than older ones is taken into account.

Macskassy and Provost (2001) measure 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 term-frequency-inverse-document frequency (TF-IDF) weights for tokens. A near neighbor text classifier (Billsus et al., 2000) uses a document vector space model. A personalized multi-document summarization and recommendation system by Radev, Fan, and Zhang (2001) recommend articles by suggesting articles from the same clusters in which the past interesting articles are located. In iScore, a variation of these methods are implemented as feature extractors in iScore. Another clustering approach, MiTAP (Damianos, Wohlever, Kozierok, & Ponte, 2003) 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

The work in iScore is closely related to the adaptive filtering task in the Text Retrieval Conference (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, relevance to a set of topics of interest is a prerequisite for interestingness. The report by Robertson and Soboroff (2002) summarize 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 relevance to a topic. Like much of the work in the task, iScore uses adaptive thresholds and incremental profile updates.

Xu et al. (2002) 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, this method is incorporated into iScore. Other methods explored in TREC11 include using a second-order perceptron, a support vector machine (SVM), a Winnow classifier (Wu et al., 2002), language modeling (Ma, Chen, Ma, Zhang, & Cai, 2002), probabilistic models of terms and relevance (Brouard, 2002), and the Okapi Basic Search System (Robertson, Walker, Zaragoza, & Herbrich, 2002).

### 2.3. Ensembles

Other work, like iScore, have leveraged multiple existing techniques to build better systems for specific tasks. For example, Henzinger (2006) combines two popular webpage duplication identification methods to achieve better results. In another example, Lazarevic and Kumar (2005) combine the results from multiple outlier detection algorithms that are applied using different sets of features.

Yan and Hauptmann (2006) combine 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. In the iScore experiments, two methods of a linear combination approach are explored, using correlation and logistic regression. But in contrast to Yan and Hauptmann (2006), functions are combined 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 the functions 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 (NIST, 2004).

Many TDT systems, like Allan, Papka, and Lavrenko (1998), Franz, Ward, McCarley, and Zhu (2001), and Allan (2002) are simply a modification of a single-pass clustering algorithm. They compare a news story against a set of profile vectors stored 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 Makkonen, Ahonen-Myka, and Salmenkivi (2004), add simple semantics of locations, names, and temporal information to the traditional term frequency vectors used in previous work.

Although a similar single-pass clustering algorithm is used in the multiple topic tracking (MTT) component of iScore, there are several subtle differences between identifying interesting articles and TDT. First, not all topics are of equal interest to an user. Instead of identifying all new topics and tracking all articles for those topics as in TDT, MTT focuses on the specific users interests, which are under continuous evolution. Additionally, MTT uses 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, MTT discards 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.

Another closely related work is in the discovery of evolutionary theme patterns (ETP) from text (Mei & Zhai, 2005). In ETP, documents are partitioned into possibly overlapping subcollections according to their publication time. The most prominent themes (or subtopics) are extracted from each subcollection. For themes in two different subcollections, an ETP solution decides whether there is an evolutionary transition from one theme to the other. The general ETP problem is not restricted to operation within an online and continuous environment, so the solution posed by Mei and Zhai (2005) is an offline data mining solution to discovering and clustering patterns and so is not directly applicable to discovering interesting articles as they are published. Additionally, the solution posed by Mei and Zhai (2005) does not learn which themes or topics are of interest to the user, and so all themes are maintained and are not useful for classifying the interestingness of an article. In ETP, the evolutionary relationships among themes at different times are also explicitly identified, which is not necessary for discovering the most interesting articles for the user as they are published.

### 2.5. Feature selection

There has been a significant amount of work done in offline feature selection. Guyon and Elisseeff (2003) survey a variety of feature selection techniques, noting cases where feature selection would improve the results of classifiers. They show that noise reduction and better class separation may be obtained by adding features that are presumably redundant. Features that are independently and identically distributed are not truly redundant. Perfectly correlated features are truly redundant in the sense that no additional information is gained by adding them. However, very high feature correlation does not mean the absence of feature complementarity. A feature that is completely useless by itself can provide a significant performance improvement when taken with others. In other words, two features that are useless by themselves can be useful together.

The feature selection method used by iScore is most similar to embedded methods, which perform feature selection during the process of training and are usually specific to given learning machines. Unfortunately, existing feature selection methods cannot be applied directly to the features used within iScore. Blum and Langley (1997) discuss feature-weighting methods such as Winnow (Littlestone, 1988). However, the inputs and outputs of the Winnow algorithm are all binary and cannot be applied directly to continuous inputs, such as the feature scores generated by iScore's feature extractors. Other Winnow variants and Winnow-based online feature selection techniques studied by Carvalho and Cohen (2006) require that all inputs are weights of importance and must be values between 0 and 1, such as normalized term frequencies. However, in general, features may not necessarily be positive weights or even have the same semantic meaning. In the case of iScore, a feature's correlation to interestingness may be positively correlated; whereas, another feature maybe negatively correlated. Utgoff, Berkman, and Clouse (1997) address this problem with 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.

Another method for feature selection is to reduce the number of redundant features, which is different from our goal of reducing the number of irrelevant features. Nurmi and Floreen (2005) identify redundant features by performing pair-wise similarities measurements using the properties of time series data, which may not be directly applied to news articles. In our experiments, we assume a more general setup, where documents from different news sources that span multiple domains are aggregated together into a single document stream and are simply ordered by publication time. Consequently, an article in the document stream is not necessarily dependent upon the content of the article that immediately precedes it in the document stream.

The feature selection method used by Xing, Jordan, and Karp (2001) employs information gain ranking and Markov blanket filtering. Although, we rank features based on their correlation to the interestingness class similar to how Xing ranks features based on their information gain, we use correlation rather than information gain due to correlation's computability in an online environment. Information gain requires the discretization of feature values which requires examining the entire range of possible values for a feature which is not possible in an online setting. The Markov blanket filtering is a more computationally intensive subset selection procedure, which is not ideal for an online setting either.

Because the importance of features for what makes an article interesting varies among users, are unknown a priori, and may change over time, no features can be discarded when constructing the overall classifier. The usefulness of each feature must be learned in an online fashion. And current online feature selection approaches are not general or efficient enough to handle the general features used by iScore.

### 2.6. Document classification

Since news articles 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 classify documents into specific topics, which is a different problem from binning documents by their interestingness.

Wong, Kan, and Young (1996) 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. Liang (2004) uses a SVM for web-page classification. Al-Mubaid and Umair (2006) use distributional clustering and a logic-based learning algorithm to classify documents. Angelova and Weikum (2006) combine the graph/network properties of documents along with traditional content classification methods to classify documents. Diaz and Metzler (2006) improve classification of documents by using multiple large external corpora. This is accomplished through a mixture of relevance models.

Latent Dirichlet Allocation (LDA) first proposed by Blei, Ng, and Jordan (2003) for document classification has been popular in document classification. LDA is a generative probabilistic model for collections of discrete data such as a 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. Newman, Chemudugunta, Smyth, and Steyvers (2006), Griffiths and Steyvers (2004), and Steyvers (2006, chapter "Probabilistic Topic Models") extend Blei et al. (2003) 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.

Yu, Zhai, and Han (2003) classify 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.

## 3. iScore pipeline

In iScore, 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 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 Fig. 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 $I(d)$:

$$I(d) = C(f_1(d), f_2(d), \ldots, f_n(d)) \tag{1}$$

Next, the adaptive thresholder thresholds the iScore to generate a binary classification, indicating the interestingness of the article to the user. Instead of using a static threshold, the threshold is dynamically adjusted. Because iScores are real numbers bounded between 0 and 1, the efficacy of every threshold between 0 and 1 in increments of 0.01 is evaluated. And in the case of ties between the utility measures, the threshold that provides the best separation between the average iScores of uninteresting and interesting articles is selected. The utility measure evaluated is FMeasure $f_\beta$, where $\beta = 0.5$, weighting precision twice as much as recall. This is consistent with the evaluation metric used in the TREC11 Adaptive Filter task (Robertson & Soboroff, 2002). This measure is discussed in further detail in the Experimental Results section.

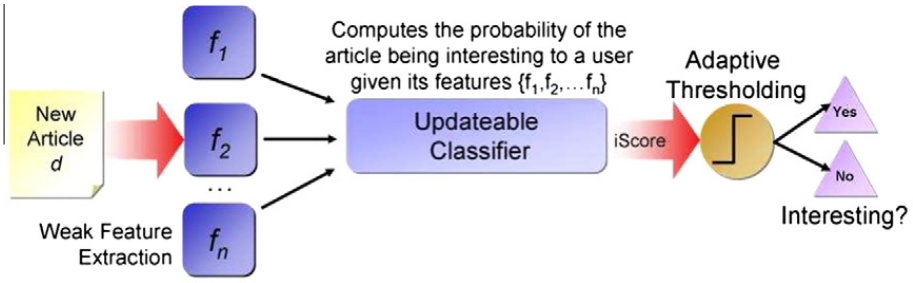


**Fig. 1.** Article classification pipeline.

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.

## 4. Features for classification

In this section, a set of article features is described that serves 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 does not determine the interestingness of an article for a user.

### 4.1. Rocchio for topic relevance

Although an article that is relevant to a topic of interest may not necessarily be interesting, relevance to such topics is a prerequisite for interestingness for a certain class of users. The standard information retrieval technique that is updateable is the Rocchio adaptive learning method (Rocchio, 1971, chapter 14). Further discussion on the Rocchio algorithm is available in Joachims (1996), in which the author compares the Rocchio relevance feedback algorithm with its probabilistic variant and the standard naïve Bayes classifier.

A document is represented as a TF-IDF vectors $\vec{d}$. The Rocchio algorithm maintains a profile vector $\vec{p}$ and updates it as follows:

$$\vec{p}_{new} = \begin{cases} \vec{p}_{old} + \chi * \vec{d} & \text{if } d \text{ is interesting} \\ \vec{p}_{old} - \gamma * \vec{d} & \text{if } d \text{ is interesting} \end{cases} \tag{2}$$

The parameters $\chi$ and $\gamma$ are the weights that determine relative weighting of positively labeled articles and negatively labeled articles, respectively. For the default configuration for Rocchio, $\chi$ is 1.0 and $\gamma$ is 0. The relevance score for the Rocchio algorithm of a document $d$ is the cosine of the angle between the profile vector and the document vector:

$$\cos(\vec{p}, \vec{d}) = \frac{\vec{p} \cdot \vec{d}}{|\vec{p}||\vec{d}|} \tag{3}$$

A variant of Rocchio by Xu et al. (2002) performed the best the TREC11 Adaptive Filter task and is used in iScore as well. It updates profiles as follows:

$$\vec{p} = \begin{cases} \vec{p} + \chi * \vec{d} & \text{if } d \text{ is interesting} \\ \vec{p} - \gamma * \vec{d} & \text{if } d \text{ is not interesting} \\ \vec{p} - \gamma' * \vec{d} & \text{otherwise and } \cos(\vec{p}, \vec{d}) < t \end{cases} \tag{4}$$

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 $\chi$, $\gamma$, $\gamma'$, and $t$ are 1, 1.8, 1.3, and 0.6, respectively (Xu et al., 2002).

### 4.2. Multiple topic tracking

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 (Schapire, Singer, & Singhal, 1998). If a user has a wide range of interests, using one vector to represent his interests would dilute the sensitivity of the Rocchio algorithm. Fig. 2 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.

Instead, with multiple topic tracking (MTT) (Pon, Cárdenas, Buttler, & Critchlow, 2007b), 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 Fig. 2, 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.

In MTT, each document and profile vector is represented as a TF-IDF vector, like in Rocchio. 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 $\vec{p}_1$ is created using the article's TF-IDF vector and added to $P$. For each subsequent article on the document stream, the closest profile in $P$ (denoted as $\vec{p}_{max}$) is compared to it. If the closeness of the new document and $\vec{p}_{max}$ is low, then a new profile is generated using the new document. Otherwise, $\vec{p}_{max}$ is updated similarily as in Rocchio.

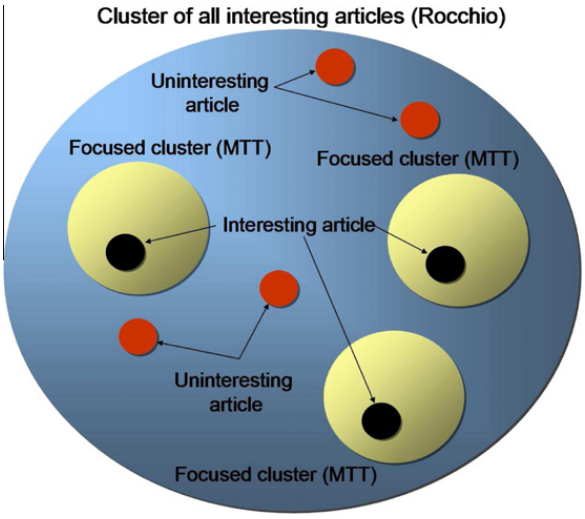


**Fig. 2.** Failure of identifying relevant documents for multiple topics.

### 4.3. eRocchio

In many information retrieval algorithms, such as the Rocchio algorithm (Rocchio, 1971, chapter 14) and MTT (Pon et al., 2007b), parameters often must be fine-tuned to a particular dataset through extensive experimentation. For example, in Xu et al. (2002), a Rocchio variant of the algorithm's performance depends extensively on the weight that is given to negatively labeled articles. In MTT, performance greatly depended upon several parameters that needed to be fine-tuned. These parameters are determined through extensive trial and error experiments. If there are many different datasets that must be evaluated, this process is often tedious and expensive, leading many to simply fine-tune the parameters to one dataset and applying the parameters globally to all other datasets, which may not be optimal. In news recommendation, user reading behavior may vary from user to user, and would result in different parameters for recommendation algorithms. For example, with regards to the weight that is applied to negatively labeled articles, one user may want to "forget" an uninteresting article relatively quickly; whereas, for another user, he may want to "forget" uninteresting articles slowly. Ideally, each user would have his own set of parameters for an algorithm like Rocchio, to identify his own set of interesting articles. This problem is even more magnified if there are many users with different reading/learning behaviors. It is not feasible for a news recommendation engine to fine-tune parameters for every user because it is very rare that validation data is available for fine-tuning until a user begins reading articles recommended by the system. Even if such a validation data was available, the task would be too time-consuming for it to be tractable if done on every user.

Given the shortcomings of existing information retrieval (IR) algorithms, such MTT, Rocchio, and its variants, that require fine-tuning parameters before the algorithms are run on live data, a different approach is taken, called eRocchio (Pon, Cárdenas, & Buttler, 2008b). Rather than predetermining the weighting scheme in the Rocchio formulation in Eq. (2), multiple instances of the Rocchio formulation are evaluated in parallel, each with a different weighting scheme. In Eq. (2), there are two unknown parameters $\chi$ and $\gamma$, the relative weights for positively labeled articles and for negatively labeled articles, respectively. However, because $\gamma$ is a weight relative to $\chi$, multiple $\gamma$-values are evaluated simultaneously while holding $\chi$ to 1.

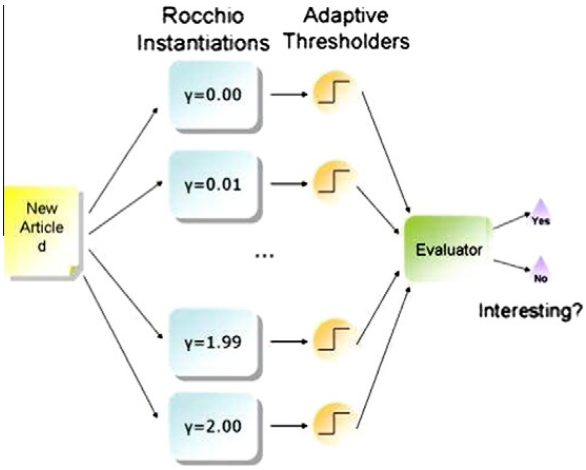


**Fig. 3.** eRocchio classification pipeline.

Each document is evaluated by multiple instantiations of the Rocchio formulation in parallel, each with a different negative article weight $\gamma$, as shown in Fig. 3. In the experimental evaluation, all possible $\gamma$-values between 0 and 2, inclusive, in intervals of 0.01, are evaluated. Because the cosine similarity between the query profile and the document is a real number bounded between 0 and 1, and a binary decision must be made, the similarity is thresholded such that articles with a high similarity with the profile are labeled as interesting and articles with low similarity are labeled as uninteresting. Rather than use a static threshold, the efficacy of every threshold between 0 and 1 in increments of 0.01 is evaluated. Each Rocchio instantiation, has its own adaptive thresholder to optimize its corresponding instantiation. Consequently, no particular distribution of interesting and uninteresting articles is assumed. And in the case of ties between utility measures, the threshold that yields the largest separation between interesting and uninteresting articles is used. Each instantiation of Rocchio has its own unique $\gamma$ and adaptive thresholder. After each adaptive thresholder has generated a binary score from its corresponding Rocchio instantiation's generated similarity score, the evaluator must generate a final answer. The best Rocchio instantiation and its corresponding threshold are chosen by selecting the Rocchio instantiation and the threshold combination that has had the best utility measure up to that point. In evaluations, FMeasure $F_\beta$ is used.

### 4.4. Language models for topic relevance

Several other methods for measuring topic relevance use language models. An $n$-gram language modeling approach has been used for document classification (Peng, Schuurmans, & Wang, 2003), which is a method that is used for finding another set of topic relevance scores. Like naïve Bayesian classifiers, language-based modeling classifiers classify documents given the number of occurrences of grams (e.g., words or characters) in the document. Unlike naïve Bayes, which assumes that grams occur independently, language modeling classifiers assume that an occurring gram is dependent upon the last $n-1$ grams. In other words:

$$P(d) = P(g_1, g_2, \ldots, g_N) = \prod_{i=1}^{N} P(g_i | g_{i-n+1}, \ldots, g_{i-1}) \tag{5}$$

where $N$ is the number of grams in the document and $g_i$ is the $i$th gram in the document $d$. $P(g_i|g_1, \ldots, g_{i-1})$ can be estimated with Jelinek–Mercer smoothing (Chen & Goodman, 1996).

In iScore, the language models are updated as new documents are processed. However, the estimation of the probabilities is time-consuming, so the number of times the model is updated is minimized while still being able to produce meaningful results. The models are compiled 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 of uninteresting articles compared to interesting ones) and to reduce compilation time, the models are updated with all interesting articles, and updated 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, several topic relevance measurements are extracted for each document. The first measurement is $P(Int|d)$, using a 6-g character model. Another measurement is $P(Int|d)$, using a uni-gram model, where grams are tokens consisting of two words – equivalent to a naïve Bayesian classifier. The third measurement is the sample cross-entropy rate between the language model of interesting past articles and the current article, using a 6-gram character model. This is commonly referred to as the binary language model classifier.

Language modeling classifiers that are trained and rebuilt in batches to reduce the consumption of computational resources often result in training delay. To address this problem, in addition to batch-building a classifier on the complete body of all articles (interesting or not) that returns a probability of whether the article is interesting, additional language models are continually built on only the title or first paragraph of interesting articles and measure the probability of the text being "generated" from the language model using Eq. (5), or the probability of the sequence of tokens in the article existing given what has been seen before. In news articles, the title and the first pargraph of an article are often very good summaries of the complete article. So by using a much smaller training text to build the models, the models are more easily updateable. But instead of computing a very large product, which can approach 0 as the number of factors to be multiplied increase, the log-probability is taken instead, transforming the product of probabilities to a sum of probabilities:

$$f_{LM}(d) = \log(P(d)) = \sum_{i=1}^{N} \log(P(g_i | g_{i-n+1}, \ldots, g_{i-1})) \tag{6}$$

For the title of the article, a language model built on bigrams, where the grams are words, is used. For the first paragraph of each interesting article, the language model built on the 6-grams, where grams are characters, is used. Using grams that are tokens for the bodies of articles would have resulted in too large and diverse of a language model to run efficiently so characters are used instead.

### 4.5. Clustering for anomaly detection

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

ing. For example, Rattigan and Jensen (2005) hypothesize that interesting articles may be produced by rare collaborations among authors. Methods for outlier detection include using mixture models (Eskin, 2000), generating solving sets (Angiulli, Basta, & Pizzuti, 2005) and using $k$-dimensional trees (Chaudhary, Szalay, & Moore, 2002), to identify outliers. In more recent work, Pilla, Loader, and Taylor (2005) propose a new statistic based on a score process for determining the statistical significance of a putative signal that may be a small perturbation in a noisy experimental background. Work in network security, such as intrusion detection has already leveraged work in outlier detection. For example, Liao and Vemuri (2002) use a k-nearest-neighbor search (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."

The first anomaly measurement used 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. At most *maxCluster* clusters are maintained, which are also represented by vectors. A count is maintained 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):

$$f_{\textit{Cluster-Anomaly}}(d) = 1.0 - \frac{\sum_{p \in P} \cos(\vec{d}, \vec{p}) \textit{size}(p)}{\sum_{p \in P} \textit{size}(p)} \tag{7}$$

After the article has been evaluated, the clusters are updated. 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 to the list of clusters given the document's vector.

### 4.6. Language models for anomaly detection

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

$$f_{\textit{LM-Anomaly}}(d) = \log(P(d|\text{documents seen before})) \tag{8}$$

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

The second language model-based anomaly detection method measures the significance and the presence of new phrases. A background model is maintained of all the documents previously seen and compare it with the language model of the current document. The sum of the significance of the degree to which phrase counts in the document model exceed their expected counts in the background model is computed. Only the top-10 phrases that exceed their expected counts are considered. A tri-gram model, where grams are word tokens, is used. Significance of each $n$-gram is based on the $z$-score (Alias, 2006):

$$z = \frac{\textit{numSuccesses} - \textit{expectedSuccesses}}{(\textit{numTrials} * P(\textit{success}) * P(1 - P(\textit{success})))^{1/2}} \tag{9}$$

with $P(\textit{success})$ defined by the $n$-gram's probability estimate in the background model, the *numSuccess* variable being the count of the $n$-gram in the foreground model, and the *numTrials* variable being the total count in in the foreground model.

The probability of the opening paragraph (or title) being "generated" from a language model of all seen first paragraphs (or titles) is measured as well. This serves as another measurement of uniqueness of an article.

### 4.7. Sliding anomaly detection

All previous measures of uniqueness examine how different the current article is with all articles seen previously. However, it may be useful to measure the uniqueness of the article with the content of news from the last $k$ days. A summary of the news from the last $k$ days (where $k$ is 30 days in the experiments) is maintained by summing the TF-IDF vectors of all articles published in the last $k$ days:

$$\vec{d}_{\textit{summary}} = \sum_{d \in \text{Articles from last } k \text{ days}} \vec{d} \tag{10}$$

The vector $\vec{d}_{\textit{summary}}$ can be incrementally maintained by keeping a history of TF-IDF vectors of the articles published in the last $k$ days and keeping a sum of the vectors as the $\vec{d}_{\textit{summary}}$ vector. The vector $\vec{d}_{\textit{summary}}$ can be updated, when necessary, by subtracting the vectors of articles published $k$ days ago and then by adding the vector of the new article. Uniqueness as measured by this measure is defined as follows:

$$f_{\textit{SlidingAnomaly}}(d) = \cos(\vec{d}_{\textit{summary}}, \vec{d}) \tag{11}$$

### 4.8. Cluster movement

Previous techniques for measuring uniqueness aim at measuring how different an article is compared to previously seen articles. Instead, another method for measuring uniqueness will be to measure the impact an article has on its parent topic by measuring how much a cluster of articles has changed when an article is added to it. Articles are added to a cluster when the TF-IDF vector that represents the centroid of the cluster is highly similar to an article (where $t \geqslant 0.4$ in the experiments). If there are no highly similar clusters, a new cluster is created using the TF-IDF vector of the article. The clustering algorithm is bootstrapped with a set of clusters to begin with.

Given the cluster that an article is closest to (if any), the change in the cluster after the article is added to the cluster is measured as follows:

$$\vec{c}_{new} = \vec{c}_{old} + \vec{d} \tag{12}$$

$$f_{ClusterMovement} = \cos(\vec{c}_{new}, \vec{c}_{old}) \tag{13}$$

### 4.9. 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 the experiments, the article's author(s) are used for the Yahoo! News and the TREC datasets, and the name of the host are used for the Digg and tagger datasets. the article's source reputation score is estimated as the average proportion of documents produced by the authors that were interesting in the past:

$$f_{Source-Rep}(d) = \frac{\sum_{a \in authors(d)} \frac{|\text{Int articles written by } a|}{|\text{Articles written by } a|}}{|authors(d)|} \tag{14}$$

In otherwords, the source reputation value is estimated as the probability of the author of the article producing an interesting article.

### 4.10. Writing style

Most work using the writing style of articles has mainly been for authorship attribution of news articles (Li, Zheng, & Chen, 2006) and blogs (Koppel, Schler, Argamon, & Messeri, 2006). Other than authorship attribution, changes in linguistic features over the course of a document have been used to segment documents as well (Chase & Argamon, 2006). Instead of author attribution and document segmentation, the same writing style features are used to infer interestingness. For example, the vocabulary richness (Tweedie & Baayen, 1998) 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.

A naïve Bayes classifier is used and trained on a subset of the features summarized by Chesley, Vincent, Xu, and Srihari (2006), including syntactic, structural, lexical, word-based, and vocabulary richness features. Like the language models used in the topic relevance measurements, the number of positive and negative articles used to update the classifier is balanced. The writing style score measured is:

$$f_{Writing\text{-}Style}(d) = P(Int|writingStyleFeatures(d)) \tag{15}$$

The writing style contribution to interestingness is estimated as the probability of the article being interesting given the writing style features.

### 4.11. 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 the temporal distance is measured between the last $k$ interesting articles and the current article:

$$f_{Freshness}(d) = \frac{1}{k} \sum_{d' \in \text{ last } k \text{ Int articles}} \log(Time(d) - Time(d') + 1) \tag{16}$$

The log of the temporal distance is measured between an interesting article and the current article since the order of magnitude in time differences is crucial. For example, an article published 1 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 ago, respectively, after the last interesting article. Essentially, freshness measures the time difference between the current article and the last set of interesting articles.

### 4.12. Topic driven freshness

The previous technique for measuring freshness measures the average log of the temporal distance of the last $k$ interesting articles with the current article. The proposition that the old freshness measure is based upon assumes that articles published closely in time with previous interesting articles are more likely to be interesting than articles published farther away in time from interesting articles. However, a better refinement of that proposition would be that articles published closely in time with the latest interesting articles in the same topic are more likely to be interesting than articles published farther away in time from the latest interesting articles in the same topic. Intuitively, topics that have not had a recent interesting article published recently are less likely to be currently interesting. Conversely, articles about topics that have had a recent interesting article are more likely to be interesting because they belong to a hot or fresh topic. Topic driven freshness is measured as follows, where the last $k$ articles in $topic(d)$ is the $k$ most recent articles that belong to the topic cluster that article $d$ is closest to:

$$f_{TDFreshness}(d) = \frac{1}{k} \sum_{d' \in \text{ Last } k \text{ articles in } topic(d)} Time(d) - Time(d') \tag{17}$$

In the experiments, $k$ is 10 articles, which seems to be a reasonable number of articles. Too large of a number would unfairly weight topics that have existed longer. Too small of a number would yield inaccurate measurements. In summary, topic driven freshness measures the time difference between the current article and the last set of interesting articles that belong to the same topic as the current article.

### 4.13. Subjectivity and polarity

The sentiment of an article may also contribute to an 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 (Mishne, 2005) and blog-specific features (Wiebe, 2000). Others have looked at subjectivity tagging, using various natural language processing (NLP) techniques (Wiebe, Wilson, Bruce, Bell, & Martin, 2004). The density of subjectivity clues in the surrounding context of a word has been used to infer its subjectivity (Wiebe, 2002) as well.

Four different features is maintained 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 Multi-Perspective Question Answering (MPQA) Opinion corpus (Wiebe, 2002) to train 6-g character language model classifiers, each sentence in the document is classified 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, the following is measured:

$$f_{class}(d) = \frac{1}{|sentences(d)|} \sum_{s \in sentences(d)} P(class|s) \tag{18}$$

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.

### 4.14. Phrase interestingness

In the techniques used in the previous sections for measuring topic relevance, bags of words are mostly used, such as in Rocchio and MTT. Language models make some effort to go beyond the bags of words approach but often examines phrases that do not make sense. For example, in the sentence,

The black dog jumped over the fence.

the following tri-grams would be examined: "The black dog," "black dog jumped," "dog jumped over," "jumped over the," and "over the fence." However, if only noun phrases are looked at, "The black dog" and "the fence" would be the only candidates. Using a noun-phrase extractor provided by OpenNLP (2006), noun phrases are extracted and normalized (making all characters in the phrase lowercase, removing stop-words, and stemming each word in the phrase). The average probability of the interestingness of noun phrases is measured as:

$$f_{PhraseInt}(d) = \frac{\sum_{p \in phrases(d)} \frac{|\text{times } p \text{ occurs in Int articles}|}{|\text{times } p \text{ occurs in articles}|}}{|phrases(d)|} \tag{19}$$

## 5. Online feature selection and 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. The ideal classifier must be able to adapt quickly to these changes.

Furthermore, the definition of interestingness varies from user to user. For example, the writing style of an article may be important for one user; whereas, for another user it may be unimportant. As a result, it is not possible to predict which features are important for a specific user before constructing the system and so all features are included for classification. Thus, classification performance suffers initially and requires a significant amount of training to adapt to the presence of useless features. And the definition of interestingness may even change for a single user over time. For example, the writing style of an article may not be important initially but may evolve to become important later on. Classifiers, such as naïve Bayes, can learn to adapt to the changing utility of features, but only with sufficient training. But because of the required long initial training period, the usefulness of the recommendation system suffers. Users of recommendation systems are less inclined to use a system if it requires a significant amount of training before it begins to give accurate recommendations. A naïve Bayes classifier is ideal for online classification since the statistics necessary for computing the probabilities are incrementally updateable. Additionally, a naïve Bayes classifier allows for the exclusion of features during classification so subsets of features can be used for classification while all features can be used for training. Thus, naïve Bayes is augmented with online feature selection (Pon, Cárdenas, & Buttler, 2008a) so that it can fulfill the role of the overall classifier adequately.

Based on Bayes' theorem, a naïve Bayes classifier assumes that features are conditionally independent (Witten & Frank, 2005). In the context of classifying articles, the probability of an article being interesting is defined by a naïve Bayes classifier as:

$$P(Int|f_1,\ldots,f_n) = \frac{1}{Z}P(Int)\prod_{i=1}^{n}P(f_i|Int) \tag{20}$$

where $Z$ is a scaling factor dependent on $f_1,\ldots,f_n$, and $Int$ is the interesting article class. The probability $P(f_i|Int)$ is estimated using kernel estimators (John & Langley, 1995). During classification, when a feature is unavailable, it is simply ignored, which is equivalent to marginalizing over them.

Ideally, an article is classified using only the most useful features for a specific user. Thus, given a set of $n$ features, the features are ordered by their current absolute Pearsons correlation to interestingness. The top-$k$ most highly correlated features for classification, where $k = 1,\ldots,n$, are considered as subset candidates. Thus, for every document, $n$ classification scores (each referred to as a subset score) are generated; one score for each subset. The overall score is the subset score associated with the subset of features with the highest FMeasure statistic. Because of the conditional independence of the features, only a single set of statistics is needed to be maintained, (in the form of kernel estimators) related to $P(f_i|Int)$ and $P(f_i)$ even though $n$ classification scores is generated for each document. For a subset of features of size less than $n$, features not in the subset are essentially ignored when generating a classification score from the naïve Bayes classifier.

After a document is classified, the classifier's kernel estimators for each feature are updated given the actual interestingness of the article. Also, the FMeasure statistic for each feature subset considered is updated as well as the correlation with interestingness for each feature.

Because statistics about each feature are continually maintained, a feature that was deemed useless earlier can be invoked for classification later. This allows for an evolving definition of interestingness for a specific user. Although irrelevant features are ignored for the overall document classification, statistics learned about the features are never forgotten.

Since only subsets of features with the highest correlations are considered for each document, as opposed to all possible subsets, this feature selection solution is tractable. Sets consisting of only features with low correlation with interestingness would be expected to be very low performing for document classification; whereas, sets of features with high correlation would be expected to be higher performing. Because only the top-$k$ most highly correlated features are considered, subsets consisting of only low correlated features are never considered. And from document to document, one would expect to see very similar top-$k$ subsets and so it may be sufficient to only update the FMeasure statistics for each top-$k$ subsets considered for that document.

The Pearsons correlation is used to evaluate the usefuless of features. Correlation is defined as:

$$correlation = \frac{E((X-\mu_X)(Y-\mu_Y))}{\sigma_X\sigma_Y} \tag{21}$$

$E$ is the expected value operator. $\mu_X$ and $\sigma_X$ are the average and standard deviaton of the random variable $X$, respectively.

It is important to note that correlation is not necessarily the best metric for measuring the utility of a feature in document classification since the actual usefulness of a feature cannot be determined by studying a single feature in isolation. There are certainly cases where two features that are useless by themselves can be useful when combined together (Guyon & Elisseeff, 2003). However, correlation is a useful guide if the features were designed to be directly or indirectly correlated with interestingness in mind, as they were for the iScore features. And by coupling this independent correlation metric with a classifier that assumes that each feature is independent, such as naïve Bayes, performance of the classifier should improve. According to (Guyon & Elisseeff, 2003), information gain and correlation are suggested for feature ranking. Information gain is difficult

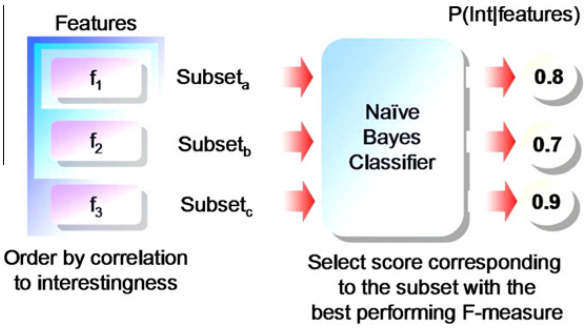


**Fig. 4.** Online feature selection with naïve Bayes.

to compute in an online fashion because the appropriate discretization is difficult to determine if the entire data is not available during evaluation (as in an online streaming environment). Consequently, correlation is used instead due to its simple online computability and its lack of a need for discretization.

For example in Fig. 4, let there be three features that are ordered in terms of correlation with interestingness: $f_1$, $f_2$, and $f_3$. Using this order, the following subsets are evaluated for their classification effectiveness: $\{f_1\}$, $\{f_1, f_2\}$, and $\{f_1, f_2, f_3\}$. If the subset $\{f_1, f_2\}$ has yielded the highest FMeasure performance so far, then it will be used to classify the next document in the document stream. After the real interestingness of the article is revealed to the system, then the classifier is updated along with the correlation of each feature with interestingness and the FMeasure of each subset. The process repeats with the next article in the document stream.

## 6. Experimental results

In this section, the recommendation performance results are summarized for the Yahoo! News, tagger, and Digg datasets, comparing the iScore with feature selection, iScore with no feature selection, the Rocchio Variant (which performed the best in the last TREC Adaptive Filter task), a language modeling classifier, MTT, and eRocchio.

### 6.1. Datasets

An interesting article is an article from a pool of articles that an arbitrary user finds interesting. The ideal dataset would consist of a large pool of articles where many individual users have provided their opinions regarding the "interestingness" of articles. Additionally, the documents in this pool would consist solely of the content of news articles (without advertising and other non-news material). However, there is currently no experimental dataset that matches the criteria perfectly so for the interesting article recommendation task, several datasets are used: Yahoo! News, tagger and Digg.

#### 6.1.1. Yahoo! news

The first dataset is a set of 35,256 and 123,653 news articles from all Yahoo! News RSS feeds (Yahoo, 2007), collected over a span of 3 months and 1 year, respectively. The smaller dataset is used in a few of the earlier experiments of iScore; whereas, the larger dataset is used for the later evaluations of iScore. The "interestingness" classification task for the Yahoo! News dataset is to identify the most interesting articles from this entire pool of articles for different communities of users. A community of users is determined by an interest-driven RSS feed from the Yahoo! News articles collection. The 43 interest-driven RSS feeds considered for labeling are feeds of the form: "Top Stories [category]," "Most Viewed [category]," "Most Emailed [category]," and "Most Highly Rated [category]," including category-independent feeds such as the "Top Stories," "Most Emailed," "Most Viewed," and "Highest Rated" feeds. 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. Note that these feeds are interest-driven and not category-driven, so the classification task is not the classical category classification task, but rather a more complex classification task. In the "interestingness" classification task, two articles that belong to the same topic may not necessarily be of equal interest to a user or a community of users.

#### 6.1.2. Tagger

In the Yahoo! News dataset, the user is modeled after a community instead of as an individual interested in a particular category. Consequently, in addition to the Yahoo! RSS feeds, the second dataset consists of articles that are anonymously collected from volunteer news readers that tag articles as they read their daily news on the web. A user can tag an article using a Firefox plug-in or a Google RSS Feed GreaseMonkey script add-on for Firefox. The script and plug-in do not record any personal identifiable information but instead uses a unique randomly generated identifier to uniquely identify users.

When a user tags an article as interesting or uninteresting, the plug-in or script records the webpage's URL, the user's tagging, and the URLs contained within the referring webpage. Articles that are pointed by links from the referring webpage that have not been read by the user are considered as uninteresting for the user since the user deemed the title of the article to be uninteresting enough to not click on. The webpages are downloaded every night. Webpages that are non-articles (e.g., advertisements, table of contents, videos) are manually removed by this author from the collection. There are only 16 users that have read and tagged at least 50 articles. The entire document collection consists of 35,656 articles. A classifier is run for each user over only the documents that have been seen by a user as indicated by a user tagging or by existing on a referring page of a tagged article.

#### 6.1.3. Digg

Due to the difficulties in recruiting volunteers to consistently read and tag articles, an alternative method is looked at, which is collected via the web service, Digg, to complement the user tagging collection. There have been 18,924 pages from the front page of Digg collected and "dugg" (Digg, 2007) by the top 100 active Digg users (Finke, 2008). Of these 100 active Digg users, 63 have "dugg" more than 100 articles from the front page, which are considered as the set of users for this dataset. With this dataset, the task is to identify which articles have been "dugg" by a user from the "Popular Stories" section of Digg. It is assumed that the most active users will be aware of the majority of the articles on the "Popular Stories" section (i.e., the front page) of Digg since the most active users on Digg are the most active users in the Digg community and are aware of at least the most popular postings. Although this actual user dataset accurately reflects individual users, the dataset is much smaller and much noisier due to the heterogeneity of the type of pages being collected. Many of the webpages downloaded may not be news but images, videos, and other webpages beyond the scope of this paper.

### 6.2. Evaluation metrics

Precision, recall, and FMeasure $f_\beta$, where $\beta = 0.5$, which weighs precision more than recall, are used for system evaluation:

$$precision = \frac{|\text{Int Articles Retrieved}|}{|\text{Articles Retrieved}|} \tag{22}$$

$$recall = \frac{|\text{Int Articles Retrieved}|}{|\text{Int Articles}|} \tag{23}$$

$$f_\beta = \frac{1 + \frac{\beta}{2}|\text{Articles Retr}|}{|\text{Articles Retr}| + \frac{\beta}{2}|\text{Int Articles}|} \tag{24}$$

In this paper, FMeasure is primarily the metric that has been focused on. FMeasure encompasses both precision and recall, so a good FMeasure score will be a balance of both of the basic retrieval metrics. FMeasure may also be defined in terms of precision and recall as follows:

$$f_\beta = \frac{(1 + \beta^2) * precision * recall}{\beta^2 * precision + recall} \tag{25}$$

FMeasure 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:

$$\begin{aligned} T11SU &= \frac{2 * \max(T11NU, -0.5) + 0.5}{1.5} \\ T11NU &= \frac{2 * |\text{Int Articles Retr}| - |\text{Unint Articles Retr}|}{2 * |\text{Interesting Articles}|} \end{aligned} \tag{26}$$

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

An ideal system would yield high FMeasure and T11SU scores overall and across time. In the experiments described in this paper, the classifiers are evaluated several different ways since no one single test nor one single dataset can definitively identify the best classifier. The follow tests are looked at:

1. The overall performance, averaged over all users in the collection, after the entire document collection is processed.
2. The overall average performance of classifiers, averaged over the bottom-$k$, the top-$k$, and all users in the collection, after the entire document collection is processed. The bottom-$k$ and the top-$k$ users are defined as the users for which iScore provided the best and worst recommendations. This test will show the performance of the classifiers for the very difficult and the very easy users to recommend for.
3. The cumulative average performance, averaged over all users in the collection, as documents are processed. This test will show any consistent overall performance differences among the classifiers, regardless of the collection size.
4. The average performance, averaged over all users in the collection, for the last 5000 documents. This test will show the current performance of the classifiers.

### 6.3. Yahoo! News

For the Yahoo! News collection, the best performing classifier seen is iScore with feature selection. Fig. 5 shows the average FMeasure, T11SU, precision, and recall of the Rocchio variant (which best performed in the TREC Adaptive Filter task), the language modeling classfier, MTT, eRocchio, iScore with no feature selection, and iScore with feature selection. The figure indicates that 0.47 FMeasure can be achieved with iScore using online feature selection, due to high precision and high recall. This is 24% better than the best baseline classifier, the language modeling classifier.

Fig. 6 shows the average FMeasure of all, the top-10 and bottom-10 performing feeds/users. The figure shows that iScore with online feature selection can give much better performance for the worst performing feeds/users than all of the other baseline classifiers. It also shows that it has the best performance of the best performing feeds/users as well.

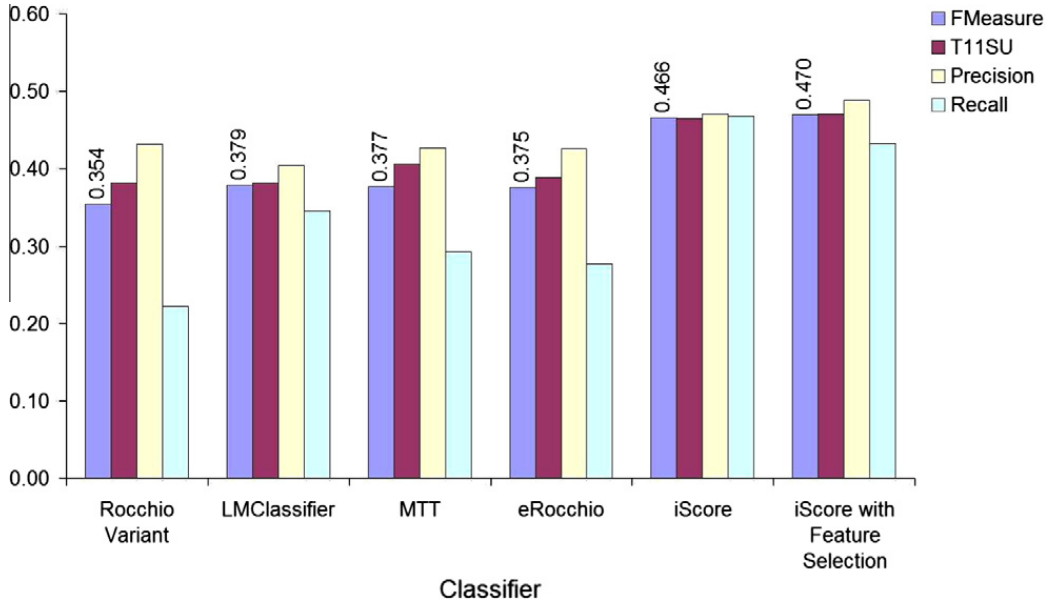


**Fig. 5.** Average performance for the Yahoo! News dataset. iScore with feature selection is 24% better than the best baseline classifiers.

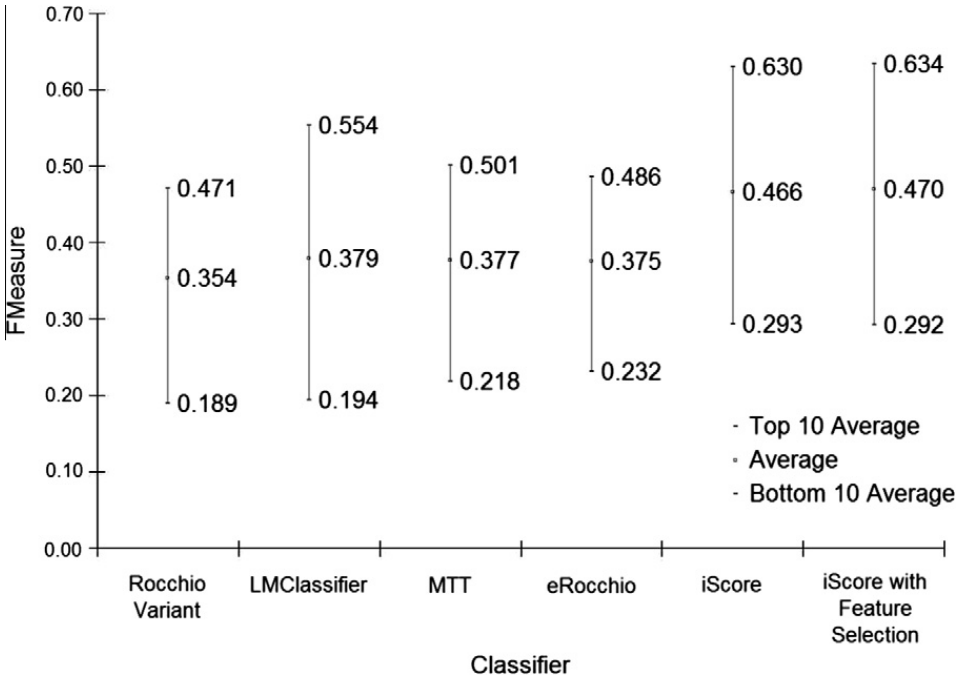


**Fig. 6.** Bottom-10, top-10, and complete average FMeasure for the Yahoo! News dataset. iScore with online feature selection can give much better performance for the average, worst, and best performing feeds/users than all of the other classifiers

A similar result is seen in Fig. 7, which shows the FMeasure performance of the classifiers over the 5000 most recent documents. The figure also shows that there is a dramatic drop in performance between periods 6 and 7 for all classifiers. This is most likely due to a long pause in the data collection for this time period.

### 6.4. Tagger

For the tagger dataset, the iScore with online feature selection performs the best, much like the Yahoo! News collection. Fig. 8 shows that very high precision and recall can achieved with iScore with online feature selection over the baseline classifiers and iScore with no feature selection.

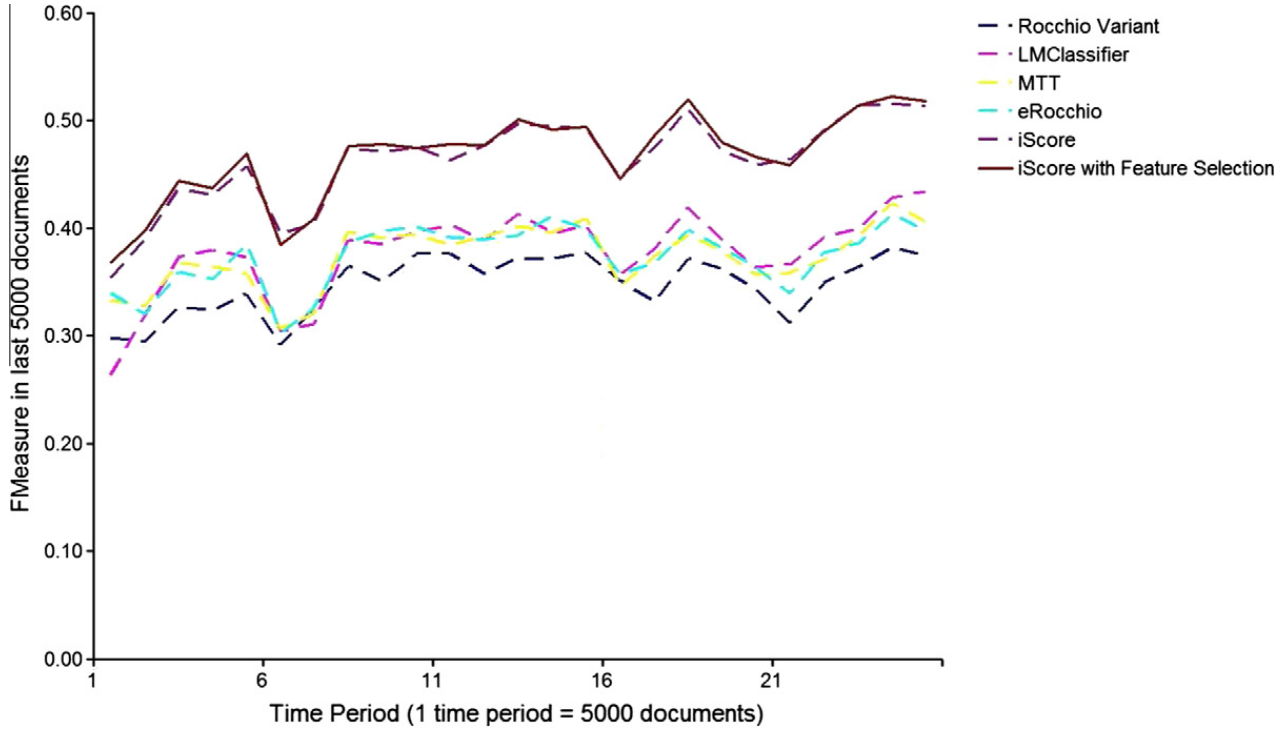


**Fig. 7.** FMeasure performance for the 5000 most recent documents for the Yahoo! News dataset.

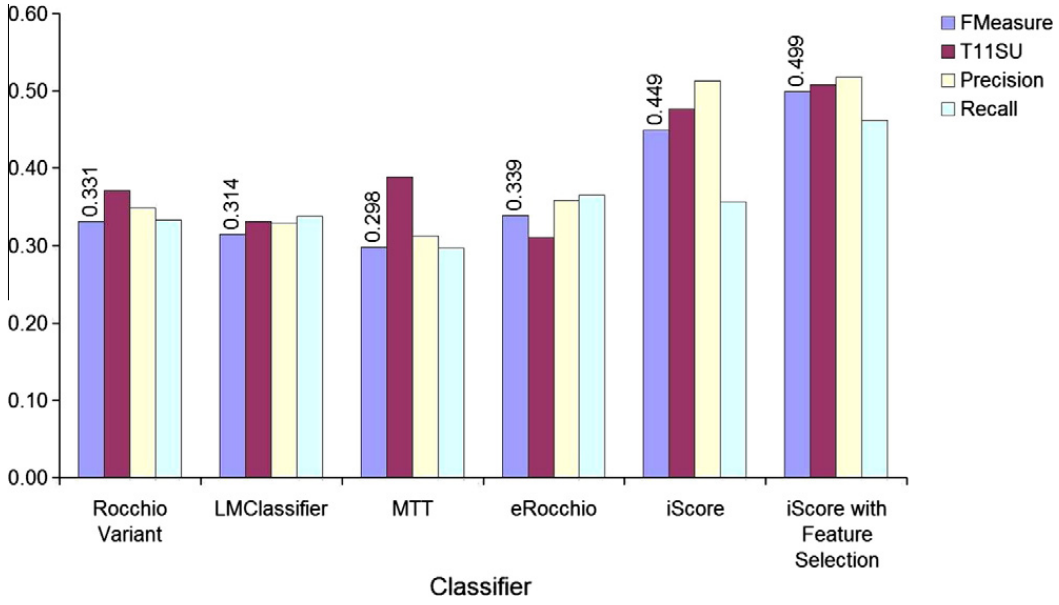


**Fig. 8.** Average performance for the tagger dataset. Very high FMeasure can achieved be with iScore with feature selection over the baseline classifiers and iScore with no feature selection.

Fig. 9 shows that iScore with online feature selection performs better due to recommendation improvements for the easiest users and most difficult users and for most users in general. The overall performance improvement provided by iScore using online feature selection over iScore with no feature selection is 11.3%. The improvement over the best baseline classifier (i.e., the Rocchio variant) is 50.7%.

### 6.5. Digg

For the Digg dataset, the best classifier seen is iScore with online feature selection. Fig. 10 shows the average FMeasure, T11SU, precision, and recall of the classifiers on the Digg data. Although, the average performance results show that eRocchio performs better in terms of FMeasure alone, iScore with feature selection has a much higher T11SU score and precision than all the other classifiers with a high FMeasure score, as shown in Fig. 10. iScore with feature selection is 2.8% better than iScore with no feature selection. Also, it is 5.2% better than the best baseline classifier (i.e., the Rocchio variant).

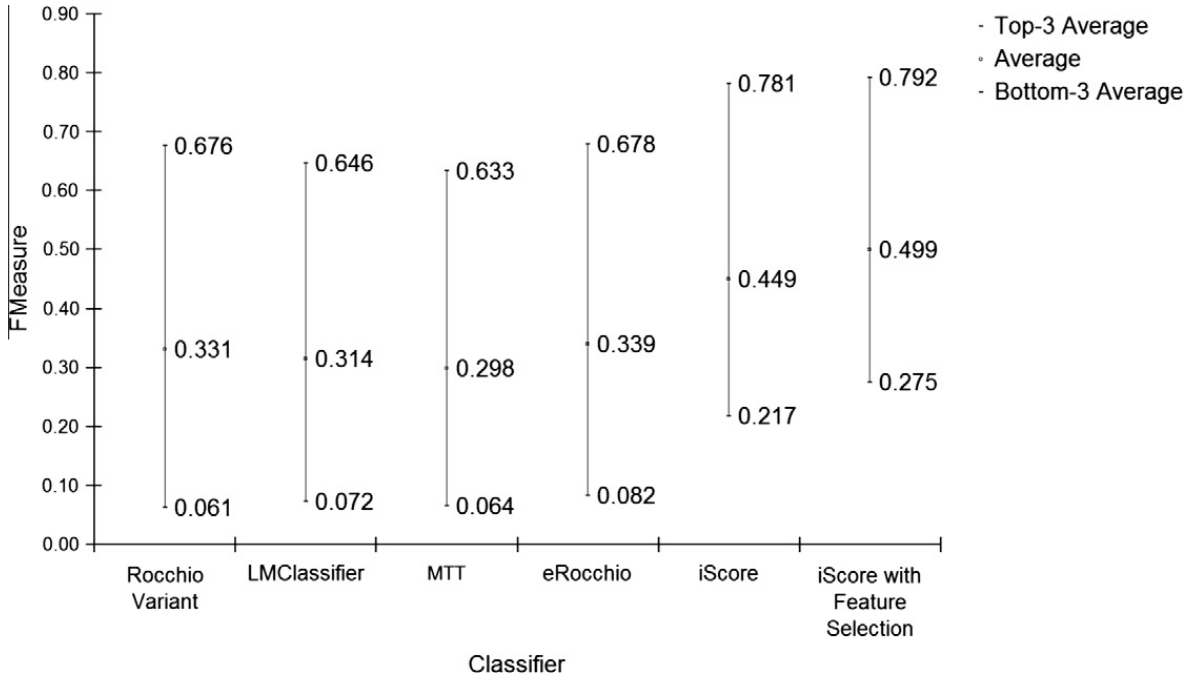


**Fig. 9.** Bottom-3, top-3, and complete average FMeasure for the tagger dataset.

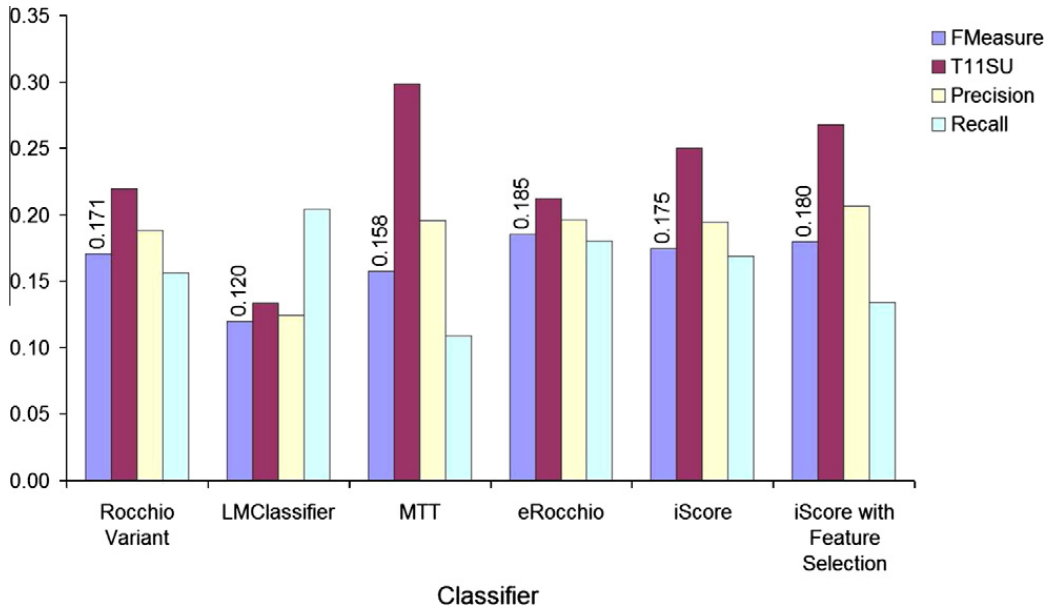


**Fig. 10.** Average performance for the Digg dataset. iScore with feature selection has a much higher T11SU score and precision than all the other classifiers with a high FMeasure score.

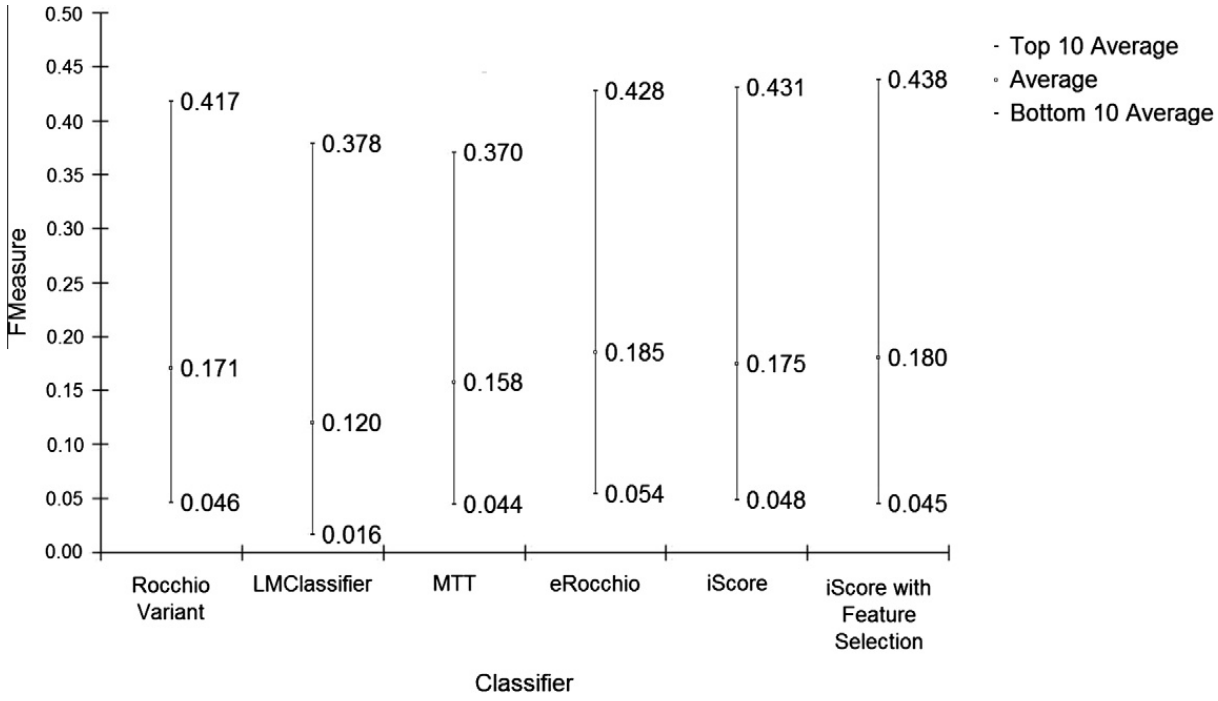


**Fig. 11.** Bottom-10, top-10, and complete average FMeasure for the Digg dataset.

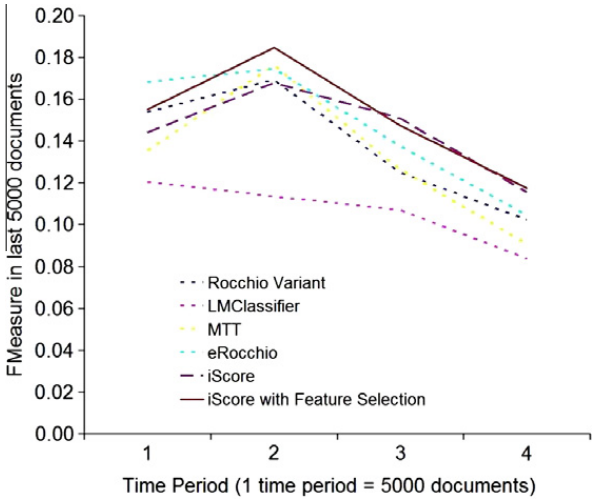


**Fig. 12.** FMeasure for the 5000 most recent documents for the Digg dataset. iScore with feature selection performs significantly better than all the other classifiers with the exception of the first time period.

Fig. 11 shows the average FMeasure of all, the easiest, and the most difficult users to recommend for. The figure indicates that for the easiest users, iScore with feature selection can recommend news articles the best. Additionally, this figure shows that online feature selection has improved upon the recommendation performance of iScore with no feature selection.

Fig. 12 shows the classifiers' FMeasure performance over the 5000 most recent documents at various time periods. The figure indicates that iScore with feature selection performs significantly better than all the other classifiers with the exception of the first time period. The figure also shows that there is less of a drop-off in performance seen in the latter time periods for iScore than the other classifiers, indicating better stability against noise and gaps in data collection.

### 6.6. Summary

In summary, iScore with online feature selection works generally well for all datasets. The Yahoo! News and tagger datasets clearly supports this hypothesis in all tests. The Digg dataset also support this hypothesis when one looks at the FMeasure performances of the classifiers over the 5000 most recent documents at various time periods. Given that all three of the datasets show the value of iScore with feature selection, it can be concluded that iScore with online feature selection oper-

ating over several high-level "interestingness"-based features can generally recommend better articles than traditional information retrieval techniques.

## 7. Conclusion

The online recommendation of interesting articles for a specific user is a complex problem, having to draw from many areas of machine learning, such as feature selection, classification, and anomaly detection. There is no single technique that will be able to address the problem of interestingness by itself. An ensemble of multiple techniques is one possible solution to addressing this problem.

To address news recommendation in a limited user environment, where methods such as collaborative filtering would perform poorly, a news recommendation framework, called iScore, is introduced to analyze the reasons why an article is interesting. The iScore framework consist of several features, including Rocchio, multiple topic tracking, online parameter selection, clustering, language models, anomaly detection, source reputation, writing style, freshness, sentiment analysis, and phrase extraction. By incorporating these features and online feature selection, iScore can generally give better recommendation results than standard information retrieval techniques with three datasets with which iScore has been evaluated: the Yahoo News!, tagger, and Digg collections. It is the combination of all these techniques that yield as much as 50.7% better recommendation results than traditional information retrieval techniques.

## Acknowledgement

This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-407783).

## References

Alias-I (2006). Lingpipe. <http://www.alias-i.com/lingpipe/index.html>.
Allan, J. (2002). Detection as multi-topic tracking. *Information Retrieval, 5*(2–3), 139–157.
Allan, J., Papka, R., & Lavrenko, V. (1998). 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* (pp. 37–45). New York, NY, USA: ACM Press.
Al-Mubaid, H., & Umair, S. A. (2006). A new text categorization technique using distributional clustering and learning logic. *IEEE Transactions on Knowledge and Data Engineering, 18*(9), 1156–1165.
Angelova, R., & Weikum, G. (2006). 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, USA: ACM Press.
Angiulli, F., Basta, S., & Pizzuti, C. (2005). Detection and prediction of distance-based outliers. In *SAC '05: Proceedings of the 2005 ACM symposium on applied computing* (pp. 537–542). New York, NY, USA: ACM Press.
Billsus, D., Pazzani, M. J., & Chen, J. (2000). A learning agent for wireless news access. In *IUI '00: Proceedings of the 5th international conference on intelligent user interfaces* (pp. 33–36). New York, NY, USA: ACM Press.
Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research, 3*, 993–1022.
Blum, A. L., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence, 97*(1–2), 245–271.
Brouard, C. (2002). Clips at TREC-11: Experiments in filtering. In *TREC11*.
Carreira, R., Crato, J. M., Goncalves, D., & Jorge, J. A. (2004). Evaluating adaptive user profiles for news classification. In *IUI '04: Proceedings of the 9th international conference on intelligent user interface* (pp. 206–212). New York, NY, USA: ACM Press.
Carvalho, V. R., & Cohen, W. W. (2006). Single-pass online learning: Performance, voting schemes and online feature selection. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 548–553). New York, NY, USA: ACM Press.
Chase, P. J., & Argamon, S. (2006). Stylistic text segmentation. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 633–634). New York, NY, USA: ACM Press.
Chaudhary, A., Szalay, A. S., & Moore, A. W. (2002). Very fast outlier detection in large multidimensional data sets. *Data Mining and Knowledge Discovery*.
Chen, S. F., & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on association for computational linguistics* (pp. 310–318). Morristown, NJ, USA: Association for Computational Linguistics.
Chesley, P., Vincent, B., Xu, L., & Srihari, R. K. (2006). Using verbs and adjectives to automatically classify blog sentiment. In *Proceedings of the AAAI-2006 spring symposium on computational approaches to analyzing weblogs*.
Corso, G. M. D., Gulli, A., & Romani, F. (2005). Ranking a stream of news. In *WWW '05: Proceedings of the 14th international conference on the World Wide Web* (pp. 97–106). New York, NY, USA: ACM Press.
Damianos, L., Wohlever, S., Kozierok, R., & Ponte, J. (2003). MiTAP: A case study of integrated knowledge discovery tools. In *HICSS '03* (p. 69).
Denning, P. J. (2006). Infoglut. *Communications of the ACM, 49*(7), 15–19.
Diaz, F., & Metzler, D. (2006). 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, USA: ACM Press.
Digg (2007). Digg. <http://www.digg.com> (September).
Eskin, E. (2000). Detecting errors within a corpus using anomaly detection. In *Proceedings of the first conference on North American chapter of the association for computational linguistics* (pp. 148–153). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
Finke, C. (2008). Diggs top 100 user. <http://www.efinke.com/digg/topusers.html> (July).
Franz, M., Ward, T., McCarley, J. S., & Zhu, W.-J. (2001). 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* (pp. 310–317). New York, NY, USA: ACM Press.
Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Science, 101*(Suppl. 1), 5228–5235. <http://dx.doi.org/10.1073/pnas.0307752101>.
Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*, 1157–1182.
Hayes-Roth, R. (2006). Two theories of process design for information superiority: Smart pull vs. smart push. In *Proceedings of the command and control research and technology symposium: The state of the art and the state of the practice*, San Diego, California.
Henzinger, M. (2006). Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 284–291). New York, NY, USA: ACM Press.
Joachims, T. (1996). A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. Tech. Rep. CMU-CS-96-118, Carnegie Mellon University.

John, G. H., & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the 11th conference on uncertainty in artificial intelligence* (pp. 338–345).
Koppel, M., Schler, J., Argamon, S., & Messeri, E. (2006). Authorship attribution with thousands of candidate authors. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 659–660). New York, NY, USA: ACM Press.
Lai, H.-J., Liang, T.-P., Ku, & Y. C. (2003). Customized internet news services based on customer profiles. In *ICEC '03: Proceedings of the 5th international conference on electronic commerce* (pp. 225–229). New York, NY, USA: ACM Press.
Lazarevic, A., & Kumar, V. (2005). Feature bagging for outlier detection. In *KDD '05: Proceeding of the 11th ACM SIGKDD international conference on knowledge discovery in data mining* (pp. 157–166). New York, NY, USA: ACM Press.
Liang, J.-Z. (2004). SVM multi-classifier and web document classification. In *Proceedings of 2004 international conference* (Vol. 3, pp. 1347–1351). Machine Learning and Cybernetics, 2004.
Liao, Y., & Vemuri, V. R. (2002). Using text categorization techniques for intrusion detection. In *Proceedings of the 11th USENIX security symposium* (pp. 51–59). Berkeley, CA, USA: USENIX Association.
Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning, 2*(4), 285–318.
Li, J., Zheng, R., & Chen, H. (2006). From fingerprint to writeprint. *Communcations of the ACM, 49*(4), 76–82.
Macskassy, S. A., & Provost, F. (2001). Intelligent information triage. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 318–326). New York, NY, USA: ACM Press.
Ma, L., Chen, Q., Ma, S., Zhang, M., & Cai, L. (2002). Incremental learning for profile training in adaptive document filtering. In *TREC11*.
Makkonen, J., Ahonen-Myka, H., & Salmenkivi, M. (2004). Simple semantics in topic detection and tracking. *Information Retrieval, 7*(3-4), 347–368.
Mei, Q., & Zhai, C. (2005). Discovering evolutionary theme patterns from text: An exploration of temporal text mining. In *KDD '05: Proceeding of the 11th ACM SIGKDD international conference on knowledge discovery in data mining* (pp. 198–207). New York, NY, USA: ACM Press.
Mishne, G. (2005). Experiments with mood classification in blog posts. In *Style2005 – The 1st workshop on stylistic analysis of text for information access, at SIGIR*.
Newman, D., Chemudugunta, C., Smyth, P., & Steyvers, M. (2006). Analyzing entities and topics in news articles using statistical topic models. In *IEEE international conference on intelligence and security informatics*.
NIST (2004). The topic detection and tracking 2004 (tdt-2004) evaluation project. <http://www.nist.gov/speech/tests/tdt/tdt2004/index.htm> (December).
Nurmi, P., & Floreen, P. (2005). Online feature selection for contextual time series data. In *PASCalforniaL subspace, latent structure and feature selection workshop*, Bohinj, Slovenia (February).
OpenNLP (2006). OpenNLP. <http://opennlp.sourceforge.net/>.
Peng, F., Schuurmans, D., & Wang, S. (2003). Language and task independent text categorization with simple language models. In *NAACL '03: Proceedings of the 2003 conference of the north american chapter of the association for computational linguistics on human language technology* (pp. 110–117). Morristown, NJ, USA: Association for Computational Linguistics.
Pilla, R. S., Loader, C., & Taylor, C. C. (2005). New technique for finding needles in haystacks: Geometric approach to distinguishing between a new source and random fluctuations. *Physical Review Letters, 95*(23), 230202. <http://link.aps.org/abstract/PRL/v95/e230202>.
Pon, R. K., Cárdenas, A. F., Buttler, D., & Critchlow, T. (2007a). iScore: Measuring the interestingness of articles in a limited user environment. In *IEEE symposium on computational intelligence and data mining 2007*, Honolulu, HI (April).
Pon, R. K., Cárdenas, A. F., Buttler, D., & Critchlow, T. (2007b). Tracking multiple topics for finding interesting articles. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 560–569). New York, NY, USA: ACM Press.
Pon, R. K., Cárdenas, A. F., & Buttler, D. J. (2008a). Improving naive Bayes with online feature selection for quick adaptation to evolving feature usefulness. In: *2008 SIAM SDM text mining workshop, Atlanta, Georgia* (April).
Pon, R. K., Cárdenas, A. F., & Buttler, D. J. (2008b). Online selection of parameters in the Rocchio algorithm for identifying interesting news articles. In *10th ACM international workshop on web information and data management (WIDM)*, Napa Valley, California (October).
Radev, D., Fan, W., & Zhang, Z. (2001). Webinessence: A personalised web-based multi-document summarisation and recommendation system. In *Proceedings of the NAACL-01* (pp. 79–88).
Rattigan, M. J., & Jensen, D. (2005). The case for anomalous link detection. In *4th Multi-relational data mining workshop, 11th ACM SIGKDD international conference on knowledge discovery and data mining*.
Robertson, S., & Soboroff, I. (2002). The TREC 2002 filtering track report. In *The 11th Text Retrieval Conference (TREC 2002)* National Institute of Standards and Technology. Available from http://trec.nist.gov/pubs/trec11/t11_proceedings.html.
Robertson, S., Walker, S., Zaragoza, H., & Herbrich, R. (2002). Microsoft Cambridge at TREC 2002: Filtering track. In *TREC11*.
Rocchio, J. (1971). *Relevance feedback in information retrieval*. Prentice-Hall (pp. 313–323).
Saracevic, T. (1996). Relevance reconsidered '96. In *Proceedings of 2nd international conference on conceptions of library and information science: Integration in perspective, Copenhagen* (pp. 201–218).
Saracevic, T. (2007). Relevance: A review of the literature and a framework for thinking on the notion in information science. *Journal of the American Society for Information Science and Technology, 58*(13), 2126–2144.
Schapire, R. E., Singer, Y., & Singhal, A., 1998. Boosting and Rocchio applied to text filtering. In *Proceedings of SIGIR-98, 21st ACM international conference on research and development in information retrieval* (pp. 215–223). New York, US, Melbourne, AU: ACM Press. <citeseer.ist.psu.edu/article/schapire98boosting.html>.
Steyvers, M. (2006). *Latent semantic analysis: A road to meaning*. Laurence Erlbaum.
Tweedie, F. J., & Baayen, R. H. (1998). How variable may a constant be? Measures of lexical richness in perspective. *Computers and the Humanities, 32*, 323–352.
Utgoff, P. E., Berkman, N. C., & Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning, 29*(1).
Wang, J., de Vries, A. P., & Reinders, M. J. T. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 501–508). New York, NY, USA: ACM Press.
Wiebe, J. (2000). Learning subjective adjectives from corpora. In *Proceedings of the 17th national conference on artificial intelligence and 12th conference on innovative applications of artificial intelligence* (pp. 735–740). AAAI Press/The MIT Press.
Wiebe, J. (2002). Instructions for annotating opinions in newspaper articles. Tr-02-101, Department of Computer Science, University of Pittsburgh.
Wiebe, J., Wilson, T., Bruce, R., Bell, M., & Martin, M. (2004). Learning subjective language. *Computational Linguistics, 30*(3), 277–308.
Witten, I. H., & Frank, E. (2005). *Data mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
Wong, J. W. T., Kan, W. K., & Young, G. (1996). Action: Automatic classification for full-text documents. *SIGIR Forum, 30*(1), 26–41.
Wu, L., Huang, X., Niu, J., Xia, Y., Feng, Z., & Zhou, Y. (2002). FDU at TREC2002: Filtering, Q&A, web and video tasks. In *TREC11*.
Xing, E., Jordan, M., & Karp, R. (2001). Feature selection for high-dimensional genomic microarray data. In *Proceedings of the eighteenth international conference on machine learning*.
Xu, H., Yang, Z., Wang, B., Liu, B., Cheng, J., Liu, Y., et al. (2002). Trec-11 experiments at Californias-ICT: Filtering and web. In *TREC11*.
Yahoo! (2007). Yahoo! News RSS feeds. <http://news.yahoo.com/rss>.
Yan, R., & Hauptmann, A. G. (2006). Probabilistic latent query analysis for combining multiple retrieval sources. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 324–331). New York, NY, USA: ACM Press.
Yu, H., Zhai, C., & Han, J. (2003). Text classification from positive and unlabeled documents. In *CIKM '03: Proceedings of the 12th international conference on Information and knowledge management* (pp. 232–239). New York, NY, USA: ACM Press.