iScore: Measuring the Interestingness of Articles in a Limited User Environment

Raymond K. Pon, Alfonso F. Cárdenas
UCLA Computer Science
420 Westwood Plaza
Los Angeles, CA 90095
(310) 825-1770
{rpon, cardenas}@cs.ucla.edu

David J. Buttler, Terence J. Critchlow
Lawrence Livermore National Laboratory
7000 East Ave
Livermore, CA 94550
(925) 422-8141
{buttler1, critchlow1}@llnl.gov

***Abstract*-Search engines, such as Google, assign scores to news articles based on their relevancy to a query. However, not all relevant articles for the query may be interesting to a user. For example, if the article is old or yields little new information, the article would be uninteresting. Relevancy scores do not take into account what makes an article interesting, which would vary from user to user. Although methods such as collaborative filtering have been shown to be effective in recommendation systems, in a limited user environment there are not enough users that would make collaborative filtering effective. We present a general framework for defining and measuring the "interestingness" of articles, incorporating user-feedback. We show 21% improvement over traditional IR methods.**

I. Introduction

An explosive growth of online news has taken place in the last few years. Users are inundated with thousands of news articles, only some of which are interesting. A system to filter out uninteresting articles would aid users that need to read and analyze many news articles daily, such as financial analysts, government officials, and news reporters. Information overload is a threat to a user's ability to function, resulting in "brain-thrashing" [1], calling for a VIRT (valued information at the right time) [2] strategy for information handling.

The most obvious approach for a VIRT strategy is to learn keywords of interest for a user [3-5]. Unfortunately, the issues related to article recommendation systems are more difficult to address than applying a simple keyword filter to weed out uninteresting articles. Although filtering articles based on keywords removes many irrelevant articles, there are still many uninteresting articles that are highly relevant to keyword searches. For example, searching for "San Francisco" in Google News will yield about 60,000 articles ordered by relevance. Unfortunately, a relevant article may not be interesting for various reasons, such as the article's age or if it discusses an event that the user has already read about in other articles.

Although it has been shown that collaborative filtering can aid in personalized recommendation systems [6], 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, we take a different approach by undertaking what makes an article interesting.

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

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

II. Related work

A. News recommendation

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

Work by [7] ranks news articles and new sources based on several properties in an online method. They claim that important news articles are clustered. They also claim that mutual reinforcement between news articles and news sources can be used for ranking, and that fresh news stories should be considered more important than old ones. In our approach, we rank news articles based on various properties in an online method, but instead of ranking articles using mutual reinforcement and article freshness, we study a different variety of features. Additionally, when training our classifiers, we also take into account that the most recent news articles are more important than older ones.

Another approach taken by [8] measures the interestingness of an article as the correlation between the article's content and the events that occur after the article's publication. For example, an article about a specific stock is interesting if there is a significant change in price after the article's publication. Using these prospective indicators, they can predict future interesting articles. Unfortunately, in most cases, these indicators are domain specific and are difficult to collect in advance for the online processing of new articles as they are published.

Other systems perform clustering or classification based on the article's content, computing such values as TF-IDF weights for tokens. A near neighbor text classifier [5] uses a document vector

This work (UCRL-CONF-225289) was performed under the auspices of the U.S. Department of Energy by the University of California Lawrence Livermore National Laboratory under contract number W-7405-Eng-48.

space model. A personalized multi-document summarization and recommendation system by [9] recommends articles by suggesting articles from the same clusters in which the past interesting articles are located. We implement a variation of these methods as feature extractors in iScore. Another clustering approach, MiTAP [10] 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.

B. *Adaptive filtering*

Our work in iScore is closely related to the adaptive filtering task in TREC, which is the online identification of news articles that are most relevant to a set of topics. The task is different from identifying interesting articles for a user because an article that is relevant to a topic may not necessarily be interesting. However, relevancy to a set of topics of interest is a prerequisite for interestingness. The report by [11] summarizes the results of the last run of the TREC filtering task. In the task, topic profiles are continually updated as new articles are processed. The profiles are used to classify a document's relevancy to a topic. We discuss some of the work in this TREC task. Like much of the work in the task, we use adaptive thresholds and incremental profile updates.

In [12], the authors use a variant of the Rocchio algorithm, in which they represent documents as a vector of TF-IDF values and maintain a profile for each topic of the same dimension. The profile is adapted by adding the weighted document vector of relevant documents and by subtracting the weighted vector of irrelevant documents. Since this approach performed the best in the task, we incorporate this method in iScore. Other methods explored in TREC11 include using a second-order perceptron, an SVM [14], a Winnow classifier [14], language modelling [15], probabilistic models of terms and relevancy [16], and the Okapi Basic Search System [17].

C. *Ensembles*

Other work, like ours, have leveraged multiple existing techniques to build better systems for specific tasks. For example, in [18], the authors combine two popular webpage duplication identification methods to achieve better results. Another example is by [19], which combines the results from multiple outlier detection algorithms that are applied using different sets of features.

A closely related ensemble work by [20] combines multiple ranking functions over the same document collection through probabilistic latent query analysis, which associates non-identical combination weights with latent classes underlying the query space. The overall ranking function is a linear combination of the different ranking functions. They extend the overall ranking function to a finite mixture of conditional probabilistic models. We explore two methods of a linear combination approach using correlation and logistic regression, but in contrast to [20], we combine functions that are not necessarily ranking functions that can be used for ranking documents for interestingness by themselves. Each function is a different aspect of interestingness and need to be combined together to generate meaningful scores for interestingness.

III. SYSTEM OVERVIEW

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.

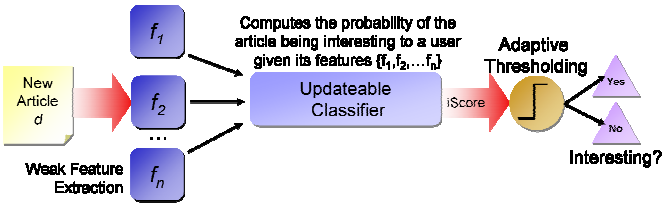

Fig. 1. Article classification pipeline

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 $I(d)$:

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

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

IV. WEAK FEATURES FOR CLASSIFICATION

In this section, we describe a set of article features that will serve as inputs into the classifier function to estimate or predict the interestingness of the article to a user. Each individual feature is a weak feature. In other words, each feature alone cannot determine the interestingness of an article for a user.

A. *Topic relevancy*

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

The first method is the Rocchio adaptive learning method [21]. Documents are represented as a vector $\vec{d}$ in a vector space. Each dimension i of the vector space represents a token t_i. The value of the vector element is the represented token's TF-IDF value. In our experiments, tokens are stems produced by the Porter algorithm [22]. Stems occurring only once in the collection are discarded to reduce the feature space, which has been shown to improve classification time and results [23].

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

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

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

The second method for measuring topic relevancy is a variant of Rocchio by [12], which updates profiles as follows:

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

The first two conditions are satisfied by user taggings. The third condition is for pseudo-negative documents, which have no taggings and its similarity with the profile is below a threshold. Good values (in TREC11) for α, β, β', and t are 1, 1.8, 1.3, and 0.6, respectively [12].

The other three methods for measuring topic relevancy use language models. An n-gram language modelling approach has been used for document classification [24], which is a method we use for finding another set of topic relevancy scores. Like naïve Bayesian classifiers, language-based modelling classifiers classify documents given the number of occurrences of grams (e.g., words or characters) in the document. Unlike naive Bayes, which assumes that grams occur independently, language modelling classifiers assume that a gram occurring is dependent upon the last n – 1 grams. In other words:

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

where N is the number of grams in the document and g_i is the i-th gram in the document d. $P(g_i|g_1,...,g_{i-1})$ can be estimated with Jelinek-Mercer smoothing [25].

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

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

B. Uniqueness

Articles that yield little new information compared to articles already seen may not be interesting. In contrast, an article that first breaks a news event may be interesting. Anomalous articles that describe a rare news event may also be interesting. For example, in [26], interesting articles may be produced by rare collaborations among authors. Methods for outlier detection include using mixture models [27], generating solving sets [28] and using k-d trees [29], to identify outliers.

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

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

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

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

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

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

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

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

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

C. 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 our experiments, we use the article's author(s). We estimate the article's source reputation score as the average proportion of documents produced by the authors that were interesting in the past:

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

D. Writing style

Most work using the writing style of articles has mainly been for authorship attribution of news articles [30] and blogs [31]. Other than authorship attribution, changes in linguistic features over the course of a document have been used to segment documents as well [32]. Instead of author attribution and document segmentation, we use the same writing style features to infer interestingness. For example, the vocabulary richness 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.

We use a naïve Bayesian classifier trained on a subset of the features from [30], including syntactic, structural, lexical, word-based, and vocabulary richness features. Like the language models used in the topic relevancy measurements, we balance the number of positive and negative articles used to update the classifier. The writing style score measured is:

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

E. 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 we measure the temporal distance between the last k interesting articles and the current article:

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

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

F. Subjectivity and polarity

The sentiment of an article may also contribute to a user's definition of interestingness. For example, "bad news" may be more interesting than "good news" (i.e., the polarity of the article). Or, subjective articles may be more interesting than objective articles. Polarity identification has been done with a dictionary [33] and blog-specific features [34]. Others have looked at subjectivity tagging, using various NLP techniques [35]. The density of subjectivity clues in the surrounding context of a word has been used to infer its subjectivity [36] as well.

We measure four different features of this feature class: polarity, subjectivity, objective speech events, and subjective speech events. A speech event is a statement made by a person, such as a quotation. Using the MPQA corpus [37] to train 6-gram character language model classifiers, we classify each sentence in the document to determine its polarity, subjectivity, and the presence of objective or subjective speech events. The MPQA corpus is a new article collection from a variety of news sources annotated for opinions and other states, such as beliefs, emotions, sentiments, and speculations. For each document and each feature in this feature set, we measure:

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

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

V. Classification

The overall classifier computes the final iScore given all the features values generated by the feature extractors. Because the features are continually refined as more documents are seen, some of the feature values may be erroneous for early documents. Also, not all the features may be useful in predicting interesting articles for a user, depending on the user's criterions. The addition of useless features has been shown to degrade the performance of classifiers [38]. Consequently, an overall classifier must be incrementally updateable, robust against noisy and potentially useless features, and generate meaningful final scores for interestingness. We evaluate four classes of classifiers: a naive Bayesian classifier, non-incremental classifiers using a sliding window, temporal inductive transfer classifiers, and a linear combination using correlation for weights.

A. Naïve Bayesian classifier

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

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

The probabilities can be estimated by maintaining statistics over feature values using kernel estimators [39].

B. Non-incremental classifiers

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

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

In this windowing approach, we first evaluate the C4.5 decision tree, built by the J48 algorithm [40]. A tree is generated using the information gain of each feature, with features with high information gain at the top of the tree and features with low information gain at the bottom of the tree. The tree is then pruned to remove branches that have low confidence in their predictive abilities; making it robust against irrelevant features.

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

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

A similar approach is taken in [20] to combine multiple ranking methods. A quasi-Newton method and ridge estimators are used to search for optimal values for λ_f [41].

In our experiments, we find that logistic regression is more accurate than C4.5 under the windowing scheme, so we evaluate logistic regression with bagging [42]. Bagging mitigates the instability of learning methods by building an ensemble of classifiers trained on randomly sampled instances from the training data. In our experiments, we build 100 ensemble classifiers.

C. Tix

We modify a method used to address concept drift, called Temporal Inductive Transfer, or Tix [43]. For every M articles processed, a new classifier is built using a base induction algorithm. The input feature vector consists of the values generated by the feature extractors along with P additional binary features. The P features are generated by predictions that the P previous classifiers would have made for the current article. To bootstrap the Tix process, the first M articles (articles in the first interval) are processed by a classifier that is continually rebuilt as new documents are read. After the first interval, the regular Tix procedure begins. In our experiments, we use logistic regression as our base induction algorithm, $P = 128$ classifiers, and $M = 1000$ articles.

D. Linear correlator

We also evaluate a linear correlator classifier that uses the correlation between a feature and interestingness. Intuitively, if a feature is highly correlated with interestingness, it should be weighted more in classifying the document. Unfortunately, it is assumed that each feature is independent, ignoring the possibility that two features that perform poorly alone in predicting interestingness may perform well when combined together [44].

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

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

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

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

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

VI. Adaptive Thresholding

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

VII. Experimental Results

iScore is implemented with an assortment of tools in Java. The system pipeline is implemented with the IBM UIMA framework [45]. LingPipe [46] is used for building language models and related classifiers. OpenNLP [47] is used for sentence detection. Other classifiers are from Weka [48].

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

The other data set comes from the TREC11 adaptive filter task,

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

We use precision, recall, and the f_β measure, where $\beta = 0.5$, which weights precision more than recall, for system evaluation. TREC11's T11SU is also used for comparing the performance of iScore with the work done in TREC11:

$$T11SU = 2 * \max(T11NU, 0.5) - 1$$

$$T11NU = \frac{2*\#\,\text{Int Articles Retr} - \#\,\text{Unint Articles Retr}}{2\times\#\,\text{Interesting Articles}} \quad (14)$$

For each data set, we use two different utility measures for guiding the thresholding. For the Yahoo! RSS articles, f_β works better than T11SU. And we find that T11SU works better for the TREC data than f_β.

A. *Feature analysis*

Fig. 2 shows the correlation of the features with interestingness in each of the RSS feeds. For most feeds, the topic relevancy and source reputation features are significantly directly correlated with interestingness. Other features, such as writing style, speech events, anomaly detection, and subjectivity have varying correlation magnitudes and directions with interestingness, depending on the RSS feed. The RSS feeds capture a variety of criterions that users may use when evaluating the interestingness of an article.

On the other hand, Fig. 3 shows that the topic relevancy and source reputation scores are the only features correlated with relevancy in the adaptive filter task. As expected, the Rocchio variant is the most correlated feature since it was the best performing filter in TREC11. Although the figure shows that the adaptive filter task captures topic relevancy well, topic relevancy is only a prerequisite for interestingness and is not sufficient for an article to be interesting. The TREC11 taggings and articles do not capture other aspects of interestingness well.

B. *Overall performance*

For the Yahoo! RSS articles, we evaluate the performance of each overall classifier against several well known topic relevancy classifiers: Rocchio, the Rocchio variant, and the 6-gram character language modelling classifier. Each classifier is coupled with the adaptive thresholding mechanism, using f_β as the utility metric. Fig. 4 shows the overall performance of the iScore classifiers compared to the baseline classifiers. The averages across RSS feeds, of precision, recall, and f_β are plotted in the graph. iScore with naïve Bayes outperforms the best baseline classifier (the language modelling classifier) by 21% in terms of f_β. iScore with the linear correlator, logistic regression, and logistic regression with bagging also perform as well as most of the baseline classifiers. iScore classifiers using Tix and the decision tree under the windowing scheme in iScore perform the worse. However, the decision tree yields high recall at the cost of precision.

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

Fig. 5 shows the performance of iScore using naïve Bayes over each of the individual feeds along with the number of articles in each feed. The feed with the worse results is the "Highest Rated Travel" feed due to the low number of articles in the feed. However, there are feeds that performed poorly despite the high number of articles in those feeds. Feeds, such as "Highest Rated" contain a variety of articles from different topics, so the topic relevancy measures, which are the most highly correlated features with interestingness overall, do not work well for these feeds.

Since iScore uses some of the methods designed for the TREC task and topic relevancy is a prerequisite for interestingness, we compare the iScore classifiers (coupled with the adaptive thresholder optimized for T11SU) with the best filters from each participating group in TREC11 in Fig. 6. Although iScore did not perform as well as the best filter, iScore with the linear correlator did perform generally well compared with most of the other filters. The next best iScore classifier is the naïve Bayesian classifier, which is consistent with the Yahoo! data set. Logistic regression, logistic regression with bagging, Tix, and C4.5 yield the worst precision and f_β score but high recall.

C. *Performance over time Periods*

Fig. 7 shows the performance of each classifier over different time periods using the Yahoo! RSS articles. Each time period contains 5000 articles. The best classifier used for iScore is the

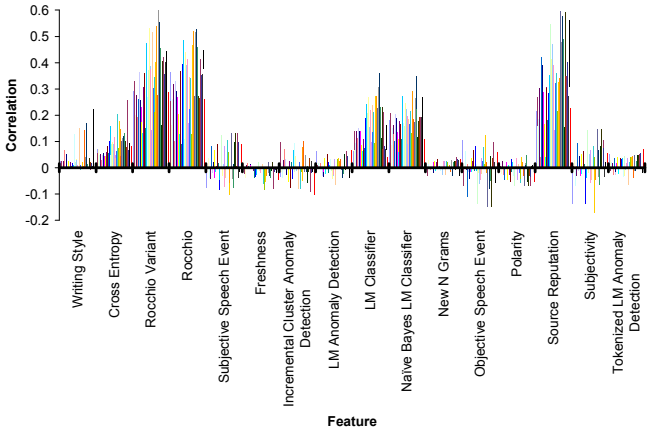

Fig. 2. Correlation of features with interestingness of the Yahoo! RSS Feed articles. Each vertical line represents an RSS feed.

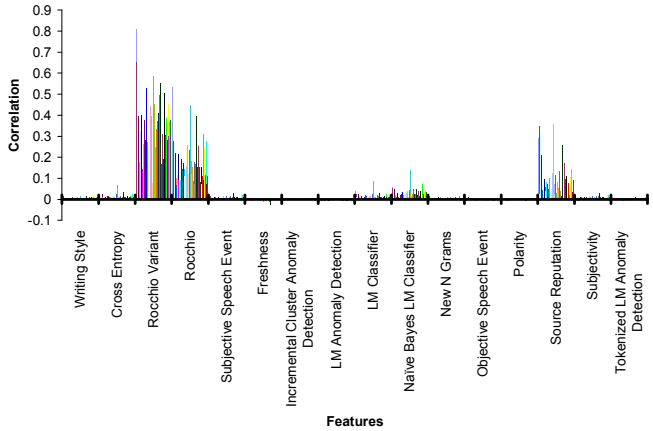

Fig. 3. Correlation of features with interestingness of the TREC11 articles and tags. Each vertical line represents a topic.

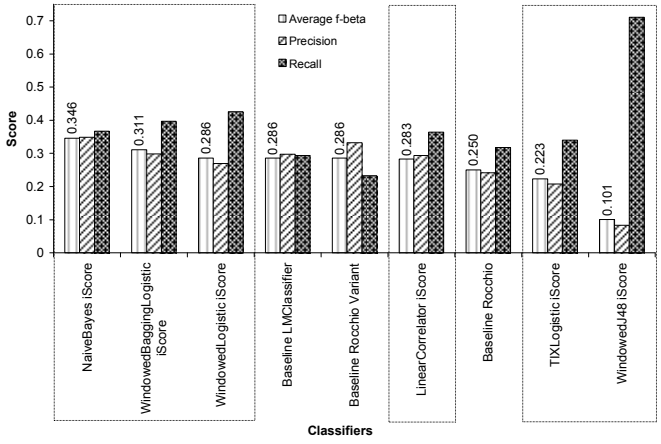

Fig. 4. Overall performance of classifiers over the Yahoo! RSS articles. The iScore classifiers are outlined.

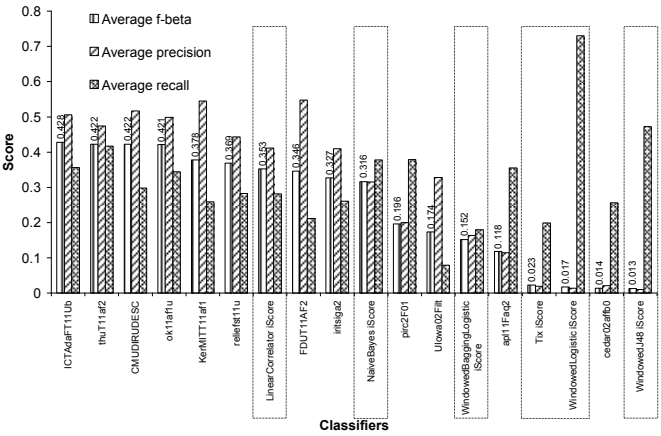

Fig. 6. Overall performance of classifiers over the TREC articles. The iScore classifiers are outlined.

naïve Bayesian classifier, outperforming the best baseline classifier (the language modelling classifier) by 24.3% on average. iScore using the naïve Bayesian classifier only performs worse than the baseline classifiers in the first time period. Because iScore has three layers of learning that must be done (i.e., the feature extractors, the overall classifier, and the adaptive thresholding); whereas, the baseline classifiers only have two layers (i.e., the classifier itself and the adaptive thresholding), iScore performs poorly at first due to propagation error among the layers. The other iScore classifiers perform generally better than the baseline classifiers with the exception of the decision tree, which fails to improve as more documents are processed.

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

We also compare the T11SU performance of iScore with the best filters from TREC11 over time in Fig. 8. Each time interval is a month's worth of articles. As in the overall performance analysis of iScore, logistic regression, logistic regression with bagging, Tix, and the decision tree perform poorly; whereas, naïve Bayes and the linear correlator perform well. In the beginning periods, naïve Bayes performs poorly compared to the TREC filters and the linear correlator, but in the latter periods, it outperforms them. The lack of overall improvement in Fig. 6 by iScore over the TREC filters and the slow increase in performance in Fig. 8 are attributed to the additional learning layers in iScore. Also the multitude of useless features for the TREC task, as shown in Fig. 3, is a contributing factor since iScore must spend more time learning which features are irrelevant. These problems can be addressed with more training, but because there are few relevant documents for each TREC topic distributed sparsely across the entire collection (proportionally much less than in the Yahoo! collection), iScore cannot immediately learn enough to outperform the other classifiers until it has seen more articles.

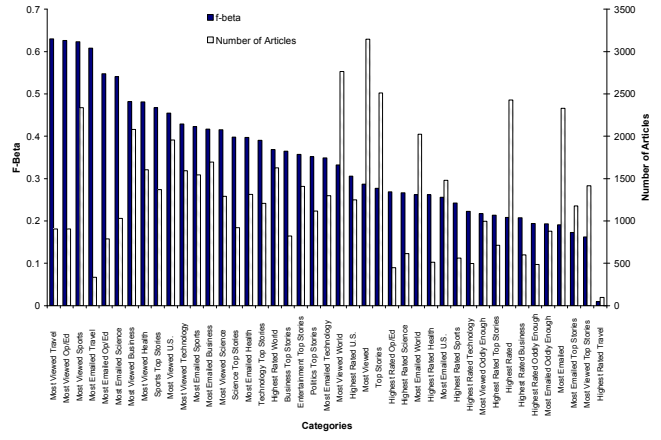

Fig. 5. Performance of iScore (using Naive Bayes) in individual categories along with the number of articles in each category.

VIII. CONCLUSION

There is room for significant improvement for personalized news recommendations systems with few users. More news articles from the Yahoo! RSS feeds are being collected, so that we can evaluate iScore over a large corpus (greater than 100,000 articles). Additionally, user taggings of articles done by volunteers using a web browser extension are being collected for further study. Also more experiments will have to be done to find optimal parameter values for the feature extractors and classifiers. Further work will also be done to address the poor results for classifying articles from general categories, such as "Highest Rated." We believe that further refinement of features other than topic relevancy may yield performance improvements for these categories. Also we believe that using named entities as well as relationships among entities can improve the accuracy of topic relevancy scoring and anomaly detection.

Unlike other personalized news recommendation systems, iScore tackles what makes an article interesting, showing that a single feature is not sufficient. Through the combination of several features using a naïve Bayesian classifier or a linear correlator, we are able to outperform most popular IR techniques in identifying interesting articles from Yahoo! RSS feeds by 21% overall and by 24.3% on average over multiple time periods. Although iScore is not specialized for retrieving articles relevant to specific topics, compared against the best filters from the TREC11 adaptive filter task, iScore performs generally well and can outperform with sufficient training.

We also show that corpora, such as TREC11, do not capture interestingness well. Such corpora only capture topic relevancy, which is only a prerequisite for interestingness. To further advance personalized news recommendation research, we call for large corpora that reflect a variety of the characteristics of interesting articles for different users.

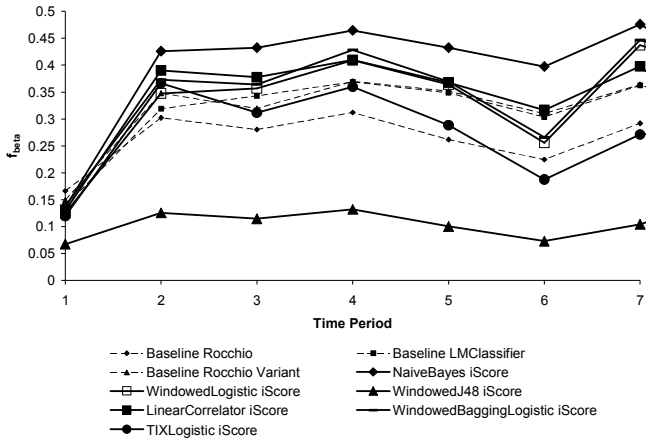

Fig. 7. Performance of classifiers over time periods over the Yahoo! RSS articles. Each time period contains 5000 articles.

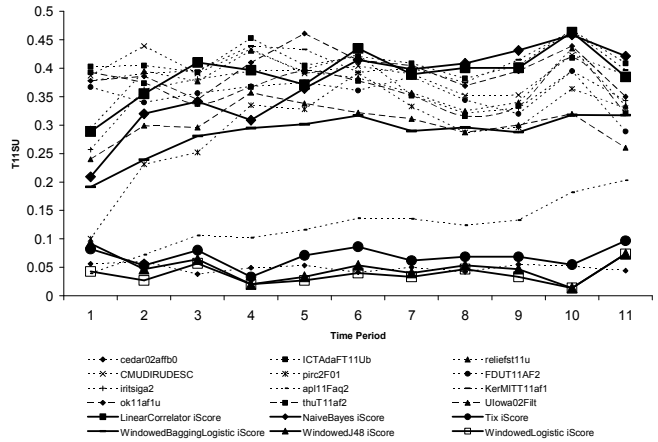

Fig. 8. Performance of classifiers over time periods over the TREC articles. Each time period is a month's worth of articles.

REFERENCES

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

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

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

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

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

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

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

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

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

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

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

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

[13] N. Cancedda, et al., "Kernel methods for document filtering," in *TREC11*, 2002.

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

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

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

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

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

[19] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceeding of the 11th ACM SIGKDD Intl. Conf. on Knowledge discovery in data mining*, 2005, pp. 157–166.

[20] R. Yan and A. G. Hauptmann, "Probabilistic latent query analysis for combining multiple retrieval sources," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006, pp. 324–331.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[43] G. Forman, "Tackling concept drift by temporal inductive transfer," in *Proc. of the 29th annual Intl. ACM SIGIR Conf. on Research and development in information retrieval*, 2006, pp. 252–259.

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

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

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

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

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