Tracking Multiple Topics for Finding Interesting Articles

Raymond K. Pon
Alfonso F. Cárdenas
UC Los Angeles
420 Westwood Plaza
Los Angeles, CA 90095
{rpon, cardenas}@cs.ucla.edu

David Buttler
Lawrence Livermore National Laboratory
7000 East Ave
Livermore, CA 94550
buttler@llnl.gov

Terence Critchlow
Pacific Northwest National Laboratory
902 Battelle Blvd
Richland, WA 99352
terence.critchlow@pnl.gov

ABSTRACT

We introduce multiple topic tracking (MTT) for iScore to better recommend news articles for users with multiple interests and to address changes in user interests over time. As an extension of the basic Rocchio algorithm, traditional topic detection and tracking, and single-pass clustering, MTT maintains multiple interest profiles to identify interesting articles for a specific user given user-feedback. Focusing on only interesting topics enables iScore to discard useless profiles to address changes in user interests and to achieve a balance between resource consumption and classification accuracy. Also by relating a topic's interestingness to an article's interestingness, iScore is able to achieve higher quality results than traditional methods such as the Rocchio algorithm.

We identify several operating parameters that work well for MTT. Using the same parameters, we show that MTT alone yields high quality results for recommending interesting articles from several corpora. The inclusion of MTT improves iScore's performance by 9% in recommending news articles from the Yahoo! News RSS feeds and the TREC11 adaptive filter article collection. And through a small user study, we show that iScore can still perform well when only provided with little user feedback.

Categories and Subject Descriptors

H3.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing, Retrieval Models, Search Process

General Terms

Algorithms, Management, Performance, Design, Experimentation, Human Factors.

Keywords

News filtering, personalization, news recommendation

1. INTRODUCTION

An explosive growth of online news has taken place in the last few years. Users are inundated with thousands of news articles, only some of which are interesting. A system to filter out uninteresting articles would aid users that need to read and analyze many news articles daily, such as financial analysts, government officials, and news reporters.

In [1], iScore is introduced to address how interesting articles can be identified in a continuous stream of news articles. Instead of applying the most naïve approach for news filtering, which is to learn keywords of interest for a user [2-4], iScore tries to identify the multitude of characteristics that make an article interesting for a specific user. In iScore, a variety of features are extracted from each article, ranging from topic relevancy to source reputation. The combination of multiple features yields higher quality results for identifying interesting articles for different users than traditional methods, such as the Rocchio algorithm [5].

Despite incorporating other article features in addition to relevancy to topics of interest, iScore still performs poorly with users that have very general interests (as opposed to very specific interests). iScore addresses relevancy by using the output of classifiers (e.g., Rocchio) that maintain a single interest profile. Unfortunately, iScore suffers when a user has a set of interests that are orthogonal to one another, which cannot be accurately represented by a single interest profile. In this paper, we extend iScore to address this shortcoming by extending the traditional Rocchio algorithm by using multiple profile vectors instead of one. This is a similar technique used in topic detection and tracking (TDT) [6], but applied to an online personalized news recommendation setting. Unlike in a TDT environment, where all new topics are identified and continually tracked by identifying their related articles, identifying interesting articles for a specific user is different for two reasons: first, not all topics are of equal interest to a user; second, a user's interest in a topic continually changes overtime. A topic that may have been interesting in the past may not be interesting in the future.

Addressing these two distinctions between TDT and news recommendation and the shortfall of the existing iScore system, we make the following contributions:

1. Instead of identifying all new topics and tracking all articles for those topics as in TDT, we focus on the specific users interests, which are under continuous evolution. Focusing on only evolving user interests instead of all topics allows for more efficient resource utilization.
2. We show that the use of multiple profile vectors yield significantly better results than traditional methods, such as the Rocchio algorithm, for identifying interesting articles. Additionally, the addition of tracking multiple topics as a new feature in iScore, improves iScore classification performance.
3. For a specific user as a case study, we analyze the operating parameters for our algorithm for their resource usage and classification performance.

4. We show that multiple topic tracking yields significantly better results than the best results from the last TREC adaptive filtering run.

2. RELATED WORK

2.1 News Recommendation Systems

In this paper, we extend iScore [1] to better handle multiple user interests. iScore is a recommendation system in a limited user environment. In addition to iScore, there are a variety of different news recommendation systems.

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 freshness by considering the most recent news articles as more important than older ones.

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 [4] uses a document vector space model. A personalized multi-document summarization and recommendation system by [8] recommends articles by suggesting articles from the same clusters in which past interesting articles are located. Another clustering approach, MiTAP [9] 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. Unlike [8] and [9], the multiple topic tracking (MTT) presented here and used by iScore clusters news articles in an online fashion as documents arrive and as the user interacts with iScore. Instead of pre-computing clusters of all documents, iScore only computes centroids of clusters of interesting articles as articles arrive on the document stream. In other words, the clusters are user-specific, as interesting articles differ from user to user. Furthermore, in MTT, each cluster, which represents a topic of interest, has an associated interestingness value which is continually updated. Using this topic interestingness and an article's relationship with the topic, MTT infers the article's own interestingness. Also, unlike other cluster-based recommendation systems, iScore's MTT metric discards uninteresting or unhelpful clusters over time to improve the quality of results and resource usage.

2.2 Adaptive Information 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 [10] 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. Like much of the work in the task, we use adaptive thresholds and incremental profile updates.

In [11], 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. Other methods explored in TREC11 include using a second-order perceptron, an SVM [12], a Winnow classifier [12], language modelling [13], probabilistic models of terms and relevancy [14], and the Okapi Basic Search System [15]. iScore's MTT, like Rocchio and [11], represents documents as vectors of TF-IDF values but instead of maintaining a single profile, MTT maintains multiple profiles to represent the distinct topics that the user is interested in and relates the topic's own interestingness to the article's interestingness.

2.3 Ensembles

Other works, like ours, have leveraged multiple existing techniques to build better systems for specific tasks. For example, in [16], the authors combine two popular webpage duplication identification methods to achieve better results. Another example is by [17], which combines the results from multiple outlier detection algorithms that are applied using different sets of features.

A closely related ensemble work by [18] 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. In contrast to [18], we combine functions that are not necessarily ranking functions in isolation. Each function is designed to capture a different aspect of interestingness and needs to be combined to generate meaningful scores for interestingness. MTT is another aspect of interestingness that is easily added to the iScore framework.

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 [6].

Many TDT systems, like [19], [20], and [21] are simply a modification of a single pass clustering algorithm. They compare a news story against a set of profile vectors kept in memory. If the story does not match any of the profiles by exceeding a similarity threshold, the story is flagged as a new event and a new profile is created using the document vector of the news story. Otherwise, the news story is used to update the existing profiles. Other work, such as [22], add simple semantics of locations, names, and temporal information to the traditional term frequency vectors used in previous work.

Although we make use of a similar single-pass clustering algorithm, there are several subtle differences between identifying interesting articles and TDT. First, not all topics are of equal interest to a user. Instead of identifying all new topics and tracking all articles for those topics as in TDT, we focus on the

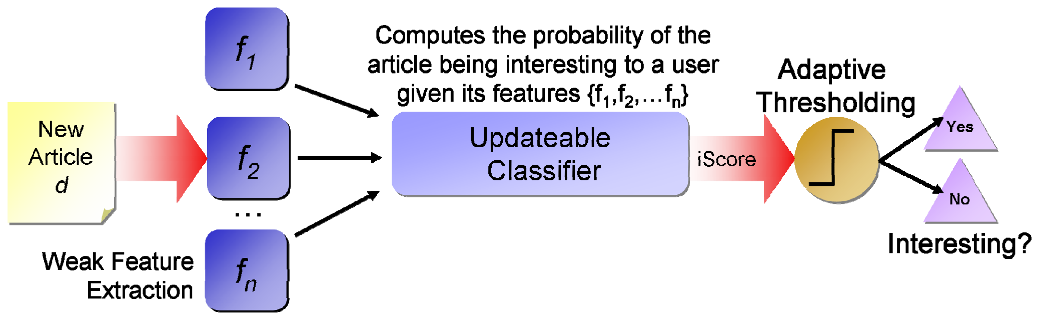


Figure 1: Article classification pipeline.

specific users interests, which are under continuous evolution. Additionally, we use the interestingness of topics when evaluating the interestingness of news articles that belong to their respective topics. Furthermore, a user's interest in a topic continually changes over time. A topic that may have been interesting in the past may not be interesting in the future. Consequently, we discard old profile vectors that are no longer of interest to reduce resource consumption, to speed up document evaluation, and to improve the quality of results.

Another closely related of work is in the discovery of evolutionary theme patterns (ETP) from text [23]. 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 any 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 [23] 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 [23] 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.

3. iScore Architecture

In iScore, news articles are processed in a streaming fashion, much like the document processing done in the TREC adaptive filter task. Articles are introduced to the system in chronological order of their publication time. Once the system classifies an article, an interestingness judgment is made available to the system by the user.

The article classification pipeline consists of four phases, shown in Figure 1. In the first phase, for an article d, a set of feature extractors generate a set of feature scores $F(d) = \{f_1(d), f_2(d),\ldots,f_n(d)\}$. In [1], we implemented several topic relevancy features, uniqueness measurements and other features, such as source reputation, freshness, subjectivity, and polarity of news articles. Then a classifier C generates an overall classification score, or an iScore $I(d)$:

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

In [1], we found that a naïve Bayesian classifier can identify interesting articles well. Next, the adaptive thresholder thresholds the iScore to generate a binary classification, indicating the interestingness of the article to the user. The adaptive thresholder tries to find the optimal threshold that yields the best metric result, such as F-measure (where β = 0.5) or TREC11's utility metric T11SU. 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. Because of iScore's extensibility, multiple topic tracking (MTT) is added to the system as a new feature extractor.

4. MULTIPLE TOPIC TRACKING

4.1 Motivation

Many information filtering algorithms are based on the Rocchio algorithm, which represents topics and documents as vectors. Each value of the vector is a TF-IDF value for its respective term [5]. A single profile vector p is maintained. For each document, the cosine similarity, or the cosine of the angle between the document vector d and the profile vector is measured.

$$\cos(d, p) = \frac{d \bullet p}{|d||p|} \tag{2}$$

The document is classified as relevant or interesting if the similarity is greater than some threshold. The profile vector is updated by adding the vector of interesting documents to the profile vector. There are variations of the Rocchio algorithm, such as subtracting irrelevant document vectors from the profile vector [11].

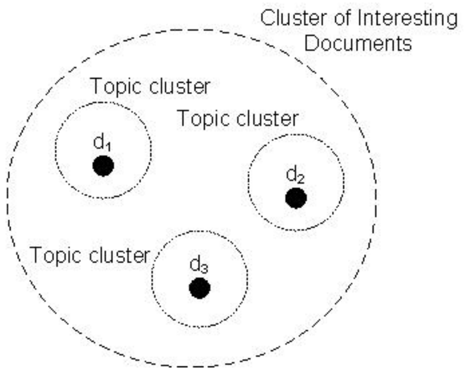


Figure 2: Failure of identifying relevant documents for multiple topics.

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 [24]. If a user has a wide range of interests, using one vector to represent his interests would dilute the sensitivity of the Rocchio algorithm. Figure 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, in MTT, a set of more narrow queries or profile vectors that more accurately represent a user's interests than a single vector is maintained. For example, in Figure 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.

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

4.2 Algorithm

In MTT, each document and profile vector is represented as a TF-IDF vector, where each value of the vector is the TF-IDF value of the vector element's corresponding stemmed term. Terms are stemmed using the Porter algorithm [25] and stop-words are ignored. A set of profiles P is maintained, which is initially empty. Until an interesting article arrives on the document stream, each article is scored with a 0. When an interesting article does arrive, a new profile vector p_1 is created using the article's TF-IDF vector and added to P. Each subsequent article on the document stream with a document vector d is processed as follows:

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

   $$f_{MTT}(d) = precision(p_{\max}) * \cos(p_{\max}, d) \quad (3)$$

   The precision of p_{max} describes how well p_{max} can accurately identify interesting articles. The precision describes how interesting the user finds the topic that the profile vector represents. By multiplying the interestingness of the topic with the document's similarity to the topic, we relate the interestingness of the containing topic to the document.
3. If the article is interesting and the similarity between d and p_{max} is less than the cluster threshold $t_{cluster}$, a new profile is generated using d. However, if the similarity is greater than or equal to $t_{cluster}$, then p_{max} is updated as follows:

   $$p_{\max} = p_{\max} + d \quad (4)$$

   Intuitively, a new profile is created because a new topic has been encountered. Each profile vector is simply the centroid of the cluster of its related articles.
4. If the article is not interesting and the similarity between d and p_{max} is greater than the classification threshold $t_{classification}$, then p_{max} is updated as follows:

   $$p_{\max} = p_{\max} - \gamma * d \quad (5)$$

   Because the profile misclassifies the article as interesting, the cluster is updated to remove the influence of terms that are not useful for predicting interestingness. This technique is similar to query zoning [26], where a select set of non-relevant articles that have some relationship to a user's interests is used for updating profile vectors. The parameter γ determines how much weight negative documents in the query zone have on the topic profile.

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

$$precision(p) = \frac{\text{\#Interesting articles w/ } \cos(p,d) > t_{classification}}{\text{\#Articles w/ } \cos(p,d) > t_{classification}} \quad (6)$$

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

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

5. EXPERIMENTAL RESULTS

iScore is implemented with an assortment of tools in Java. The system pipeline is implemented with the IBM UIMA framework [27], using classifiers from LingPipe [28], OpenNLP [29], and Weka [30].

We evaluate iScore against three 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. The

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

The final 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 serve as reasonable proxies for a set of users.

We use precision, recall, and F-measure, where $\beta = 0.5$, which weights precision more than recall, for system evaluation:

$$F_\beta = \frac{\left(1+\frac{\beta}{2}\right)|\text{Int Articles Retr}|}{|\text{Articles Retr}| + \frac{\beta}{2}|\text{Int Articles}|} \quad (7)$$

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

$$T11SU = 2*\max(T11NU, 0.5) - 1$$
$$T11NU = \frac{2*|\text{Int Articles Retr}| - |\text{Unint Articles Retr}|}{2*|\text{Interesting Articles}|} \quad (8)$$

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

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

5.1 Case Study: Operating Parameters

To find good values for the operating parameters: $t_{cluster}$, $t_{classification}$, $t_{precision}$, γ, and M, we evaluate the resource consumption and the quality of results produced by MTT with various parameters for the "Politics Top Stories" RSS feed. For simplicity and to evaluate MTT in isolation, we use the pipeline shown in Figure 3 instead of the complete iScore pipeline. The adaptive thresholder optimizes for F-measure ($\beta = 0.5$).

We first evaluate the effect of $t_{cluster}$ on MTT by varying $t_{cluster}$ while holding $t_{precision}$, $t_{classification}$, M, and γ at 0.5, 0.5, ∞., and 0, respectively. The results are shown in Figure 4a. As $t_{cluster}$ increases, articles are discouraged from clustering. Consequently, the average number of profiles held increases as $t_{cluster}$ increases. Low $t_{cluster}$ values cause fewer clusters to be formed, causing MTT to behave similarly as Rocchio when $t_{cluster}$ is low. The figure also shows that any $t_{cluster}$ value greater than 0.6, there is no significant increase in the quality of results while there is an increase in resource consumption. From this observation, we use 0.6 for $t_{cluster}$ for all subsequent experiments.

Next we evaluate the effect of $t_{precision}$ on MTT by varying $t_{precision}$ while holding $t_{cluster}$, $t_{classification}$, M, and γ at 0.6, 0.5, ∞., and 0, respectively. Figure 4b shows that there is little variation in performance overall when $t_{precision}$ is varied. However, there is a slight increase in performance when $t_{precision}$ increases from 0.3 to 0.5. There is also a decrease in resource consumption in the same range. Overall, fewer profile vectors are kept in memory but the profiles are more precise in identifying interesting articles when $t_{precision}$ is increased. Performance peaks when $t_{precision} = 0.5$, with performance slightly decreasing for any values beyond 0.5 with very little decrease in resource consumption. Consequently, for all later experiments, we use 0.5 for $t_{precision}$.

Next we evaluate $t_{classification}$ on MTT by varying $t_{classification}$ while holding $t_{cluster}$, $t_{precision}$, M, and γ at 0.6, 0.5, ∞., and 0, respectively. As $t_{classification}$ increases, the average number of profiles held in memory increases. Profiles are discarded when they generate too many false positives due to the minimum precision discard mechanism. Higher $t_{classification}$ values make it more difficult for the profiles to generate false positives (while generating many more false negatives), so fewer profiles are discarded. Figure 4c shows that a good value for $t_{classification}$ is 0.4. Values less than 0.4 cause profiles to generate too many false positives and are consequently discarded, so fewer profiles are kept, decreasing the number of topics that can be tracked. However, too high of a value for $t_{classification}$ will lead to too many false negative classifications, as shown in the decrease in recall in Figure 4c. Interestingly, there is also a decrease in precision for too high $t_{classification}$ values. This is most likely due to noise caused by anomalous taggings that would normally be removed when low precision profiles are discarded. It is also due to changes in user interests which are immediately addressed by removing low precision profiles as well. For all subsequent experiments, we use 0.4 for $t_{classification}$..

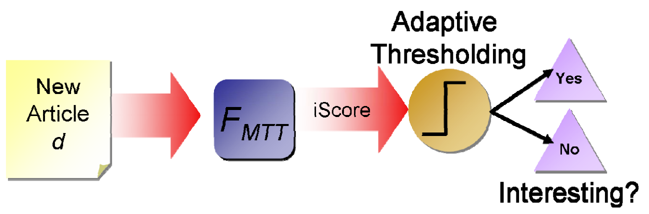
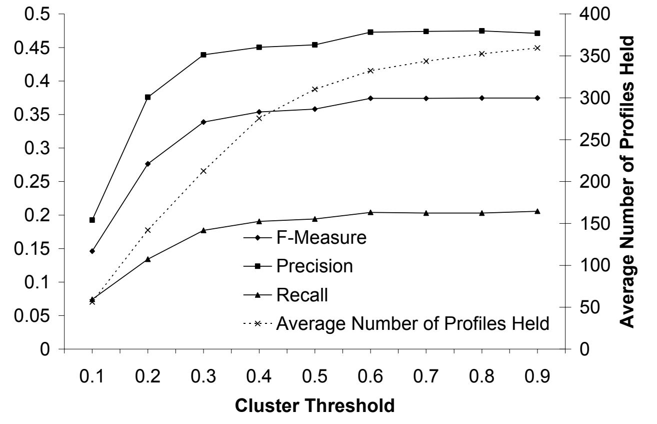
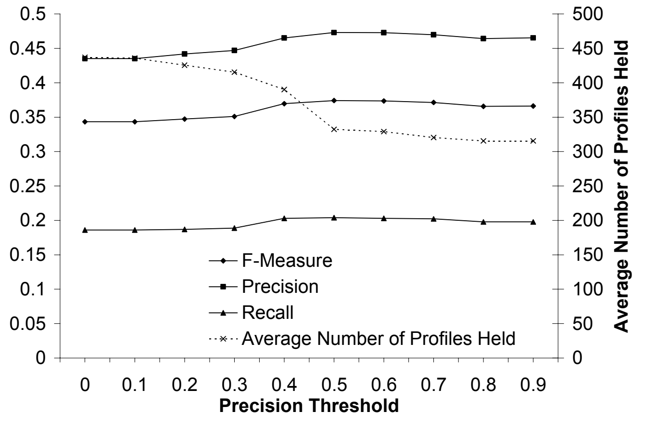


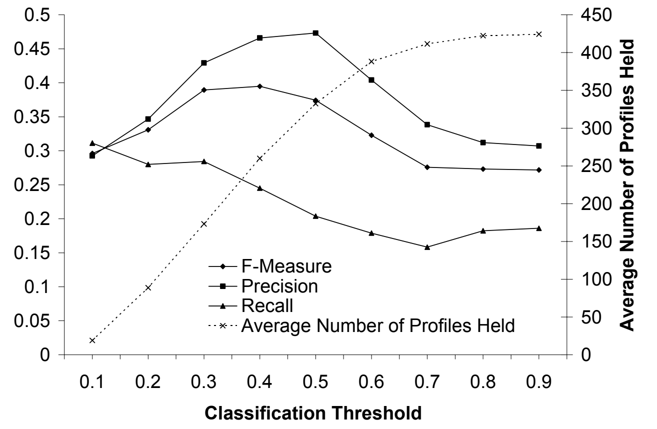
Figure 3: MTT evaluation pipeline.



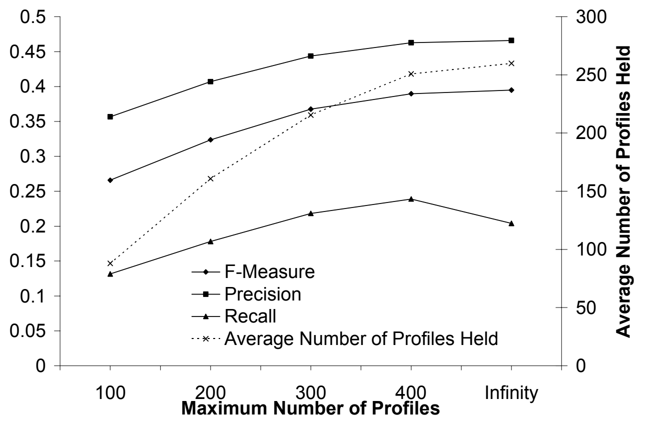
(a)



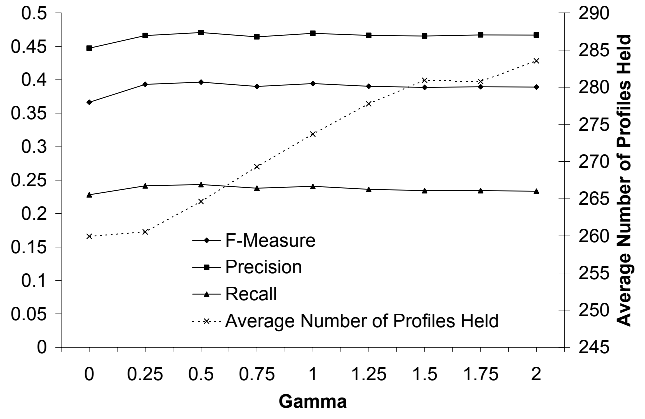
(b)



(c)



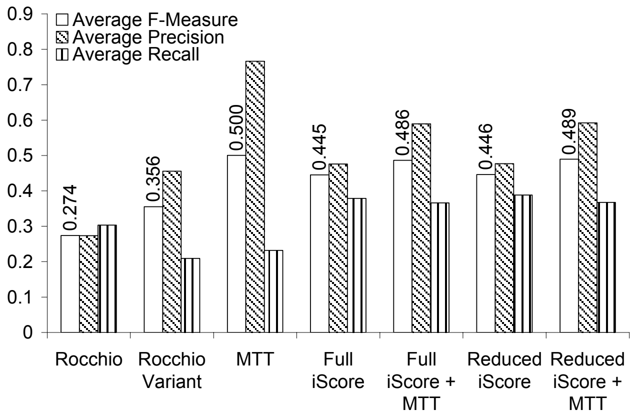
(d)



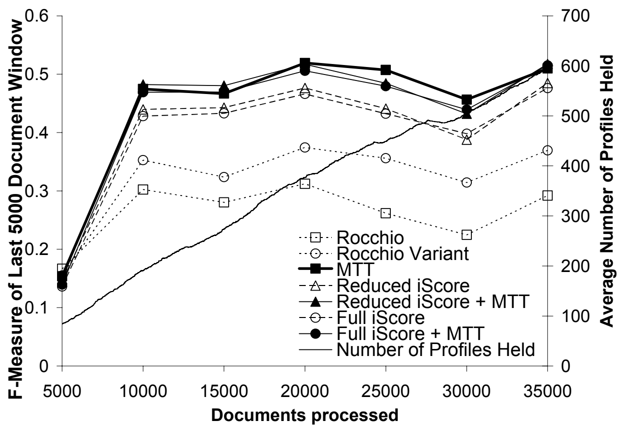
(e)

Figure 4: Operating parameters for Politics Top Stories from the Yahoo! RSS Feeds

Next we evaluate the effect of *M*, which is the maximum number of profiles kept in memory, while holding $t_{cluster}$, $t_{precision}$, $t_{classification}$, and γ at 0.6, 0.5, 0.4, and 0, respectively. Figure 4d shows that that as the number of maximum profiles increase, the quality of results improve, with significantly higher precision. However, it is inconclusive to determine if it is better to leave the number of profiles unbounded because the number of profiles kept in memory for "Politics Top Stories" is at most 500. It is difficult to determine the tradeoff in resource consumption with performance, given the results in Figure 4d. A data set that spans a large period of time is likely needed to determine a good *M* value. So for all subsequent experiments, we leave the number of profiles kept in memory to be unbounded, or $M=\infty$.



(a)



(b)

Figure 6: Performance over time of iScore and MTT using the Yahoo! RSS Feeds. Figure 5a shows the overall performance of the classifiers after processing 10,000 documents. Figure 5b shows the performance of the classifiers over time.

Finally, we evaluate the effect of γ, which controls how much of an effect that misclassified uninteresting articles have (compared to interesting articles), by varying γ, while holding $t_{cluster}$, $t_{precision}$, $t_{classification}$, and M at 0.6, 0.5, 0.4, and ∞., respectively. Figure 4e shows that there is very little effect caused by γ. There is only a slight increase in performance when γ =0.5. However, there is a significant change in memory consumption, as γ increases. Since the profiles are actually the centroids of clusters of interesting documents, using a large γ value, changes the natural document clusters. As more documents are processed, new clusters must be generated since the natural clusters no longer exist. However, using a small non-zero γ value can help reduce the noise in the clusters and improve the classification quality. For all subsequent experiments, we use 0.5 for γ.

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

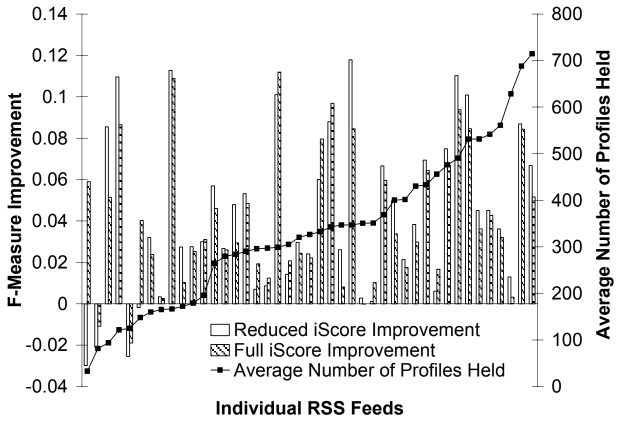


Figure 5: Overall improvement (after processing 10,000 documents) of iScore by including MTT for individual RSS feeds. Each column is an individual RSS feed.

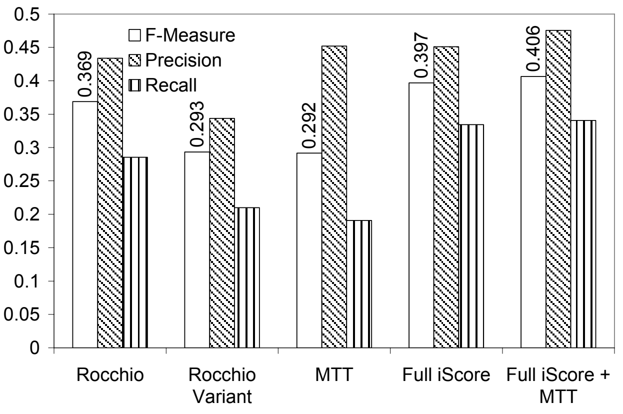
5.2 Overall Performance

Given the results from the case study for finding good operating parameters for the "Politics Top Stories" RSS feed, we applied MTT with the same parameters to all the other RSS feeds. In this experiment as well, the adaptive thresholder optimizes for F-measure. We use $t_{cluster}$=0.6, $t_{precision}$=0.5, $t_{classification}$=0.4, M=∞, and γ =0.5. The mean average results are shown in Figure 6.

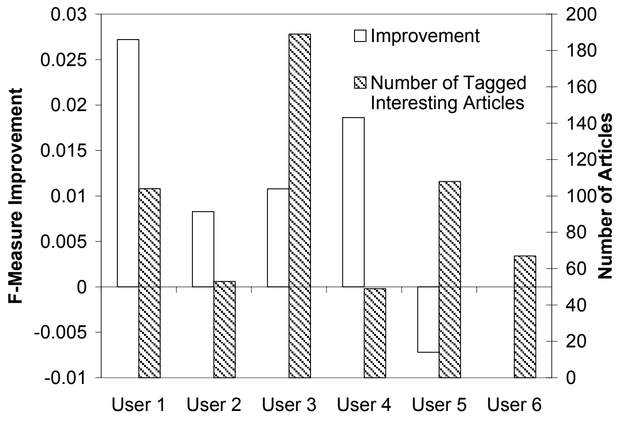
In Figure 6a, we compare the overall performance of various classifiers after processing 10,000 documents, including Rocchio and the Rocchio variant from [11], which performed the best in the TREC11 adaptive filter task. The figure shows that MTT performs significantly better than the Rocchio variant, with a mean average F-measure (where β=0.5) of 0.500, performing 40% better. MTT also outperforms Rocchio by 82%. According to the t-test, MTT's improvements over Rocchio and its variant are statistically significant (p = 2.7E-11 over Rocchio, p = 1.6E-06 over the Rocchio variant).

We also evaluate MTT when it is included in the complete iScore pipeline. We evaluate two iScore systems, one with the complete feature set from [1] and one with a reduced feature set with the highest correlated features to interestingness. The reduced feature set contains all the features from [1] except for freshness, new n-gram anomaly detection, and n-gram and tokenized language modelling anomaly detection. Figure 6a shows iScore with MTT has a similar F-measure performance as MTT alone, with iScore yielding greater recall and lower precision. Figure 6a also shows that the inclusion of MTT into iScore results in a statistically significant increase of 9% in F-measure (p = 0.09). Closer examination of each individual RSS feed shows positive improvement for most RSS feeds in Figure 5. RSS feeds with a lower average number of profiles held in memory due to very few interesting articles are more likely to yield negative improvement.

Figure 6b shows the performance of the classifiers over time as well as the mean average number of profiles held in memory. The F-measure dramatically increases after processing 10,000 articles. The figure also shows statistically significant improvements ($p <$ 0.02) that MTT and iScore with MTT have over Rocchio, the Rocchio variant, and iScore without MTT. After processing 25,000 documents, there is a statistically significant 10% increase in performance of iScore caused by the inclusion of MTT (p =



(a)



(b)

Figure 7: Performance of iScore and other classifiers in the user tagging collection. Figure 7a shows the overall performance of the classifiers. Figure 7b shows the overall of iScore by including MTT for individual users along with the number of tagged articles for each user.

0.02), which is consistent to what was observed in Figure 6a. Although, it seems that MTT alone performs better than iScore with MTT after processing 25,000 documents, the t-test shows that that increase is not statistically significant ($p = 0.23$) for both feature sets. The figure also shows that the average number of profiles held increases linearly as more documents are processed. This behavior is expected since new topics continually appear and the maximum number of profiles is left unbounded for these experiments so no unused interesting topics are discarded. Further study with a larger corpus is necessary to determine a good maximum, if any.

5.3 User Study

In addition to the Yahoo! RSS feed articles, we also compare the performance of MTT and iScore with Rocchio and the Rocchio variant on a collection of web pages tagged by users using a web browser plug-in. In our experiments where positive user taggings are sparse, we find that the adaptive thresholder performs better when it optimizes for T11SU instead of F-measure, so in this set of experiments, the adaptive thresholder optimizes for T11SU. Also due to the scarcity of negative user taggings compared to the size of the entire data collection, we infer additional negative user taggings for articles that the user did not read but were accessible from the referring page of an article tagged by the user. Consequently, for each user, interesting news articles are predicted from a pool of articles consisting of articles that the user actually tagged and articles that were accessible from referring pages of tagged articles. We use the same operating parameters found in our case study for the “Politics Top Stories” RSS feed.

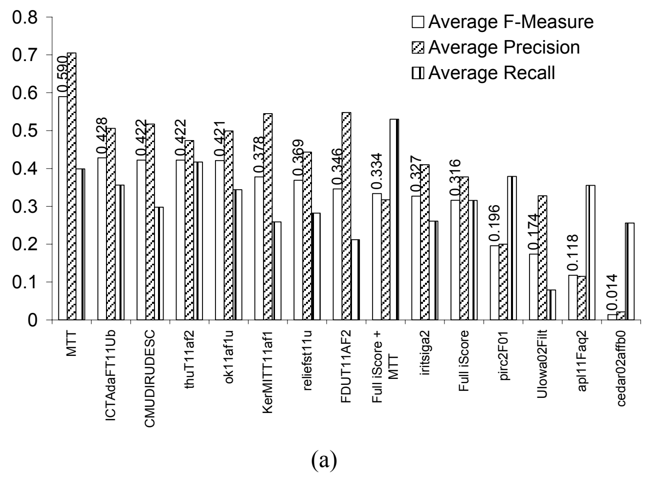
The results of the user study are shown in Figure 7. In Figure 7a, MTT has higher precision than the Rocchio variants but has much lower recall. As a result, MTT’s F-measure performance is lower than the Rocchio variants. iScore with MTT performs 10% better than Rocchio in terms of F-measure. And the inclusion of MTT improves iScore by 2%. However, the comparison of the results is difficult to determine because the t-test indicates that the comparisons are not statistically significant with $p \geq 0.25$. More users are necessary to make a definitive judgment, which is likely to be similar to the statistically significant judgments made with the Yahoo! data set. Closer inspection of each individual user shows F-measure improvement to iScore for most users when MTT is included into iScore’s feature set, as shown in Figure 7b.

Although a smaller and potentially noisier data set, this limited user study shows that iScore with MTT can work well even with a limited number of user taggings. For the Yahoo! data set, each RSS feed has an average of 655 tagged articles. In contrast, this limited study, as shown in Figure 7b, has a much smaller collection of taggings with each user tagging an average of 95 interesting articles.

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

5.4 TREC Filtering

Although the TREC11 adaptive filter task is to retrieve all articles relevant to a query, regardless of its interestingness to a user, we want to see how well MTT and iScore performs against other adaptive filters from TREC11. MTT and the full iScore feature set are compared with the best filters from each participating group in TREC11 against the TREC11’S RCV1 corpus in Figure 8. As in the user tagging collection, the taggings in the TREC collection are very sparse relative to the size of the collection, so the adaptive thresholder optimizes for T11SU as well. We use the same operating parameters found in our case study for the “Politics Top Stories” RSS feed. Figure 8a shows that MTT has an F-measure 37.8% better than the best performing filter in TREC11 [11]. MTT also yields higher precision and recall. When MTT is incorporated into iScore, F-measure improves by 9% but is not statistically significant ($p = 0.25$). According to [1], features other than topic relevancy features are not useful for identifying interesting articles to the TREC topics. The addition of irrelevant features causes the statistically significant difference in performance between MTT alone and iScore with MTT ($p < 0.01$)



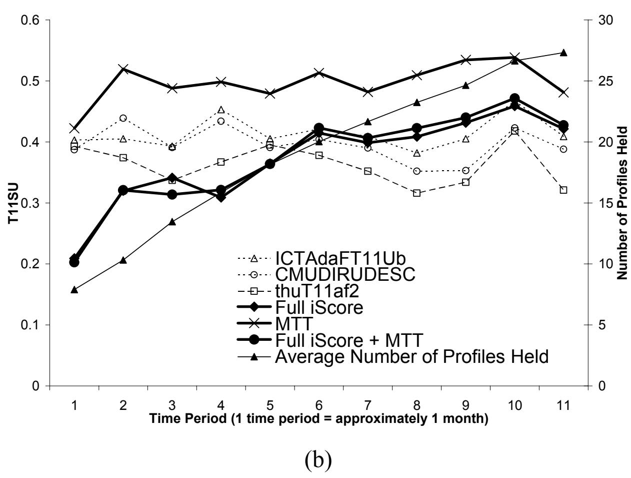


Figure 8: Performance of iScore and MTT in the TREC11 adaptive filter task. Figure 8a shows the overall performance of the classifiers. Figure 8b shows the performance of the classifiers over time.

while only improving iScore slightly when added as an additional feature.

Figure 8b shows the performance of iScore and MTT along with the top three adaptive filters from Figure 8a. TREC only reports T11SU performance over time instead of F-measure, so T11SU is shown in Figure 8b. The figure shows that MTT performs much better over time than all the other classifiers. Like Figure 8a, Figure 8b does shows statistically insignificant improvement of 2% ($p = 0.31$) for documents after time period 6.

Comparing Figure 6b and Figure 8b, the resource consumption in the TREC data is much less than in the Yahoo! data. Despite the larger size of the TREC corpus, user interests in the TREC data set are much narrower than that of the Yahoo! data so fewer profile vectors are necessary to represent the entire range of user interests.

6. DISCUSSION AND FUTURE WORK

Interestingly, in the Yahoo! data set, MTT alone yields statistically similar F-measure performance as iScore using the implemented features from [1] along with MTT. MTT alone may be sufficient if precision is desired over recall. But if a balance of recall and precision is needed, then iScore with MTT would be better than MTT alone. Also, there are use cases where MTT may be insufficient. MTT excels at identifying new interesting articles for topics that have already been seen. However, MTT would fail at identifying “flash point” articles that discuss new interesting topics that are unrelated to the previously interesting topics. On the other hand, the iScore framework allows for future features to be added to the classification pipeline, such as those that would help identify “flash point” articles, in addition to MTT as another feature.

The inclusion of MTT into iScore has improved the classification of interesting documents for most users and data sets by 9% overall. However, there is room for further study. More news articles from the Yahoo! RSS feeds are being collected, so that iScore can be evaluated over a larger corpus (greater than 100,000 articles). In our case study, due to the size of the Yahoo! RSS feed used to evaluate MTT and the number of relevant articles in the TREC11 adaptive task, the resource usage behavior of MTT when the maximum number of profiles is varied could not be accurately determined. A larger Yahoo! RSS feed collection spanning a large time period would help determine a good value, if any. More user taggings of articles by volunteers are being collected to improve the quality of the user tagging data set as well.

Additionally, in the presented experiments, the parameters were static for all users. In our case study, it was shown that two different users can have two different near-optimal parameters configurations. Using the parameter configuration found here as starting points and given the expected behavior of MTT for various parameters, optimal parameters tailored for specific users with specific memory constraints and quality requirements can be dynamically learned as more documents are processed so that maximum performance for each user can be achieved with MTT.

7. CONCLUSION

Multiple Topic Tracking (MTT), inspired by the Rocchio algorithm and single-pass clustering algorithms used in topic detection and tracking, is shown to be an effective technique to classifying news articles as interesting or not interesting for specific users. By explicitly and distinctly tracking multiple topics of user interest and their degree of interestingness, MTT addresses the shortcomings of the Rocchio algorithm’s usage of a single query to find all interesting articles from across multiple topics and its inability to quickly adapt to changes in user interests.

Through a case study for a single RSS feed, we found reasonably good operating parameters for MTT. Using these parameters, MTT and iScore with MTT is able to perform 40% to 82% better than existing Rocchio variants when recommending interesting articles from the Yahoo! News RSS feeds. For the TREC adaptive filter task, MTT overall performs 37.8% better the best adaptive filter from TREC11. MTT also outperforms the same filter over time as more documents are processed in terms of TREC11’s T11SU metric. The inclusion of MTT can improve iScore’s performance by 9% overall.

Although more users are necessary to make a more definitive conclusion on the performance of MTT and iScore in our small user study, iScore with MTT seems to outperform Rocchio and its variant. Also through our limited user study, we show that iScore can still work relatively well even with very few positively tagged articles.

8. ACKNOWLEDGMENTS
This work (UCRL-CONF-228286) 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.

9. REFERENCES

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

[2] R. Carreira, J. M. Crato, D. Gongalves, and J. A. Jorge, "Evaluating adaptive user profiles for news classification," in *IUI '04: Proceedings of the 9th international conference on intelligent user interface*. New York, NY, USA: ACM Press, 2004, pp. 206-212.

[3] H.-J. Lai, T.-P. Liang, and Y. C. Ku, "Customized internet news services based on customer profiles," in *ICEC '03: Proceedings of the 5th international conference on Electronic commerce*. New York, NY, USA: ACM Press, 2003, pp. 225-229.

[4] D. Billsus, M. J. Pazzani, and J. Chen, "A learning agent for wireless news access," in *IUI '00: Proceedings of the 5th international conference on intelligent user interfaces*. New York, NY, USA: ACM Press, 2000, pp. 33-36.

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

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

[7] G. M. D. Corso, A. Gulli, and F. Romani, "Ranking a stream of news," in *WWW '05: Proceedings of the 14th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2005, pp. 97-106.

[8] 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

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

[10] S. Robertson and I. Soboro, "The TREC 2002 filtering track report," in *TREC 2002*, 2002.

[11] H. Xu, Z. Yang, B. Wang, B. Liu, J. Cheng, Y. Liu, Z. Yang, X. Cheng, and S. Bai, "TREC-11 experiments at CAS-ICT: Filtering and web," in *TREC11*, 2002.

[12] 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.

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

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

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

[16] M. Henzinger, "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. New York, NY, USA: ACM Press, 2006, pp. 284-291.

[17] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. New York, NY, USA: ACM Press, 2005, pp. 157-166.

[18] R. Yan and A. G. Hauptmann, "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. New York, NY, USA: ACM Press, 2006, pp. 324-331.

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

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

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

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

[23] Q. Mei and C. Zhai, "Discovering evolutionary theme patterns from text: an exploration of temporal text mining," in Proceedings of the 11th ACM SIGKDD international conference on knowledge discovery in data mining. New York, NY, USA: ACM Press, 2005, pp. 198-207.

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

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

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

[27] IBM, "Unstructured information management architecture SDK," Website, 9 2006. [Online]. Available: http://www.alphaworks.ibm.com/tech/uima

[28] Alias-I, "LingPipe," Website, 9 2006. [Online]. Available: http://www.alias-i.com/lingpipe/index.html

[29] OpenNLP, "OpenNLP," Website, 9 2006. [Online]. Available: http://opennlp.sourceforge.net/

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