Online Selection of Parameters in the Rocchio Algorithm for Identifying Interesting News Articles

Raymond K. Pon
UC Los Angeles
420 Westwood Plaza
Los Angeles, CA 90095
rpon@cs.ucla.edu

Alfonso F. Cárdenas
UC Los Angeles
420 Westwood Plaza
Los Angeles, CA 90095
cardenas@cs.ucla.edu

David J. Buttler
Lawrence Livermore National Laboratory
7000 East Ave
Livermore, CA 94550
buttler1@llnl.gov

ABSTRACT
We show that users have different reading behavior when evaluating the interestingness of articles, calling for different parameter configurations for information retrieval algorithms for different users. Better recommendation results can be made if parameters for common information retrieval algorithms, such as the Rocchio algorithm, are learned dynamically instead of being statically fixed *a priori*. By dynamically learning good parameter configurations, Rocchio can adapt to differences in user behavior among users. We show that by adaptively learning online the parameters of a simple retrieval algorithm, similar recommendation performance can be achieved as more complex algorithms or algorithms that require extensive fine-tuning. Also we have also shon that online parameter learning can yield 10% better results than best performing filter from the TREC11 adaptive filter task.

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. Although it has been shown that collaborative filtering can aid in personalized recommendation systems [1], 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 so recommendation systems must rely solely on the content of the articles in such environments.

In [2], we introduced iScore to address how interesting articles can be identified in a continuous stream of news articles. In iScore, a variety of information retrieval algorithms are used to identify interesting articles. However, in many information retrieval algorithms, such as the Rocchio algorithm [3], parameters often must be fine-tuned to a particular data set through extensive experimentation. For example, in [4], a Rocchio variant of the algorithm's performance depends extensively on the weight that is given to negatively labeled articles. This parameter is determined through extensive trial and error experiments. If there are many different data sets that must be evaluated, this process is often tedious and expensive, leading many to simply fine-tune the parameters to one data set and applying the parameters globally to all other data sets, which may not be optimal.

In news recommendation, user reading behavior may vary from user to user, and would result in different parameters for recommendation algorithms. For example, with regards to the weight that is applied to negatively labeled articles, one user may want to "forget" an uninteresting article relatively quickly; whereas, for another user, he may want to "forget" uninteresting articles slowly. Ideally, each user would have his own set of parameters for an algorithm like Rocchio, to identify his own set of interesting articles.

This problem is magnified if there are many users with different reading/learning behavior. It is not feasible for a news recommendation engine to fine-tune parameters for every user because it is very rare that validation data is available for fine-tuning until a user begins reading articles recommended by the system. Evenif such validation data was available, the task would be too time-consuming for it to be done on every user.

To address this problem in news recommendation, we make the following contributions:

1. We show that users have different learning/reading behavior when evaluating the interestingness of news articles.
2. Instead of using static parameters, we show that by evaluating several different parameter configurations simultaneously, better recommendation and retrieval

WIDM'08, October 30, 2008, Napa Valley, California, USA.
Copyright 2008 ACM 978-1-60558-260-3/08/10...$5.00.

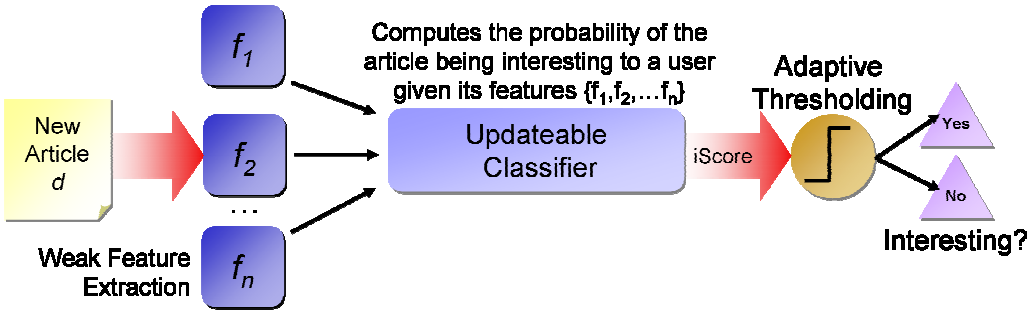

Figure 1. iScore article classification pipeline.

results can be achieved.

3. We show that the performance gains are due to different user behavior.
4. We show that by learning the parameters of a simple information retrieval algorithm online, we can attain similar performance as more complex information retrieval algorithms, such as language modelling classifiers, and algorithms that require fine-tuning, such as Rocchio.

2. RELATED WORKS

2.1 Rocchio

The Rocchio algorithm, first introduced in [3], models documents and queries as TF-IDF vectors. It aims at forming the optimal query so that documents that are highly similar to the query are marked as relevant. When applied to adaptive document filtering, the query is continually updated. In general, the query profile $\vec{Q}$ is updated as the following:

$$\vec{Q}_{new} = \alpha * \vec{Q}_{orig} + \chi * \sum_{D \in Rel} \vec{D} - \gamma * \sum_{D \notin Rel} \vec{D} \qquad (1)$$

The parameters χ and γ represent the weights when adding positive and negatively tagged articles to the query profile. The χ parameter represents rate of emphasizing the terms of positively tagged articles. The γ represents the rate of deemphasizing terms from negatively tagged articles. The vector $\vec{D}$ is the TF-IDF vector of an article. The set *REL* is the set of all relevant or positively tagged articles. The vector $\vec{Q}_{new}$ is the TF-IDF vector of the query profile. The vector $\vec{Q}_{orig}$ is the TF-IDF vector of some search query string. In a text filtering setting, there is often no initial user-query so $\alpha * \vec{Q}_{orig}$ is ignored [5], simplifying the Rocchio formulation to the weighted sum of relevant documents and irrelevant documents:

$$\vec{Q}_{new} = \chi * \sum_{D \in Rel} \vec{D} - \gamma * \sum_{D \notin Rel} \vec{D} \qquad (2)$$

The Rocchio formulation can be incrementally computed as the following:

$$\vec{Q}_{new} = \begin{cases} \vec{Q}_{old} + \chi * \vec{D} & \text{if } D \in Rel \\ \vec{Q}_{old} - \gamma * \vec{D} & \text{if } D \notin Rel \end{cases} \qquad (3)$$

All negative components of the resulting profile are assigned a zero value. A document is classified by Rocchio as relevant if its cosine similarity with the query profile is above a threshold. The cosine similarity between a document with a vector $\vec{D}$ and a query profile $\vec{Q}$ is defined as:

$$\cos(\vec{D}, \vec{Q}) = \frac{\vec{D} \bullet \vec{Q}}{|\vec{D}||\vec{Q}|} \qquad (4)$$

Other variations on Rocchio include the use of query zoning [6] where only the set of non-relevant documents considered for the profile update are those that relate well to the user's interest (i.e., have high similarity to the query profile). Another variation makes the distinction between soft negative articles (i.e., unlabeled articles that are not relevant to the query) and hard negative articles (i.e., labeled articles that are not relevant to the query). For example, [7] uses different weights for negatively labeled documents and unlabeled documents. In [7], Rocchio is further extended using many more parameters, including the use of multiple query profiles to represent the multiple interests of a single user. In that algorithm, called MTT, the optimal set of parameters may vary from user to user, depending on the users' interests.

However, the problem with these Rocchio variants is that the weighting schemes for the Rocchio formulation must be predetermined ahead of time. Often, this requires fine-tuning the parameters for the specific query and for the corpus. By pre-setting the parameters, it is assumed that the tuned parameters are the optimal ones for all users, which may not necessarily be the case.

Other works have looked at Rocchio from a theoretical point of view. For example, in [8], the lower bound of the number of mistakes that Rocchio will make in different scenarios was studied. In [9], the connection between Rocchio and probabilistic classifiers, such as naïve Bayes, was identified.

2.2 iScore

Some of the Rocchio variants have been adapted into iScore [2]. The Rocchio variant discussed in this study can also be incorporated into iScore. iScore aims to accurately predict interesting news articles for a single user. News articles are processed in a streaming fashion, much like the document processing done in the TREC adaptive filter task [10]. Articles are introduced to the system in chronological order based on their publication time. Once the system classifies an article, an

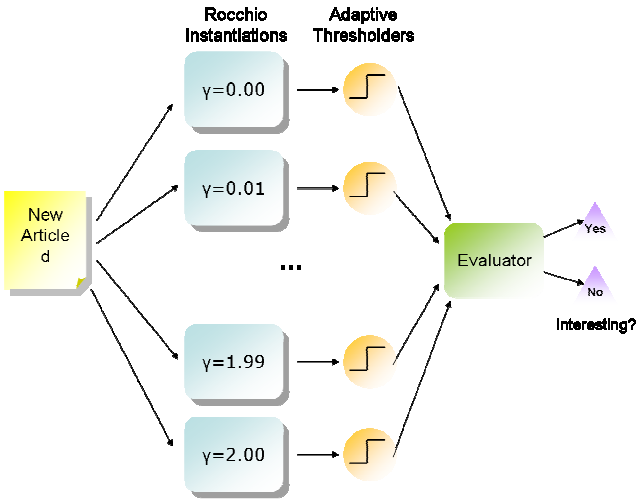

Figure 2. eRocchio evaluation pipeline.

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 [2], several topic relevancy features, uniqueness measurements and other features, such as source reputation, freshness, subjectivity, and polarity of news articles were implemented. Then a classifier C generates an overall classification score, or an iScore $I(d)$:

$$I(d) = C(f_1(d), f_2(d), \ldots, f_n(d)) \quad (5)$$

In [2], we found that a naïve Bayesian classifier has acceptable performance in identifying interesting articles. Next, an adaptive thresholder thresholds the iScore to generate a binary classification that indicates whether or not the article is interesting to the user. The adaptive thresholder tries to find the optimal threshold that yields the best metric result, such as F-Measure. 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, new recommendation algorithms, such as the one discussed here, can be added to the system as a new feature extractor.

3. eROCCHIO

Given the shortcomings of existing IR algorithms such MTT and Rocchio and its variants that require fine-tuning parameters before the algorithms are run on live data, we have taken a different approach. Rather than predetermining the weighting scheme in the Rocchio formulation in Equation 3, multiple instances of the Rocchio formulation are evaluated in parallel, each with a different weighting scheme. In Equation 3, there are two unknown parameters χ and γ, the relative weights for positively labeled

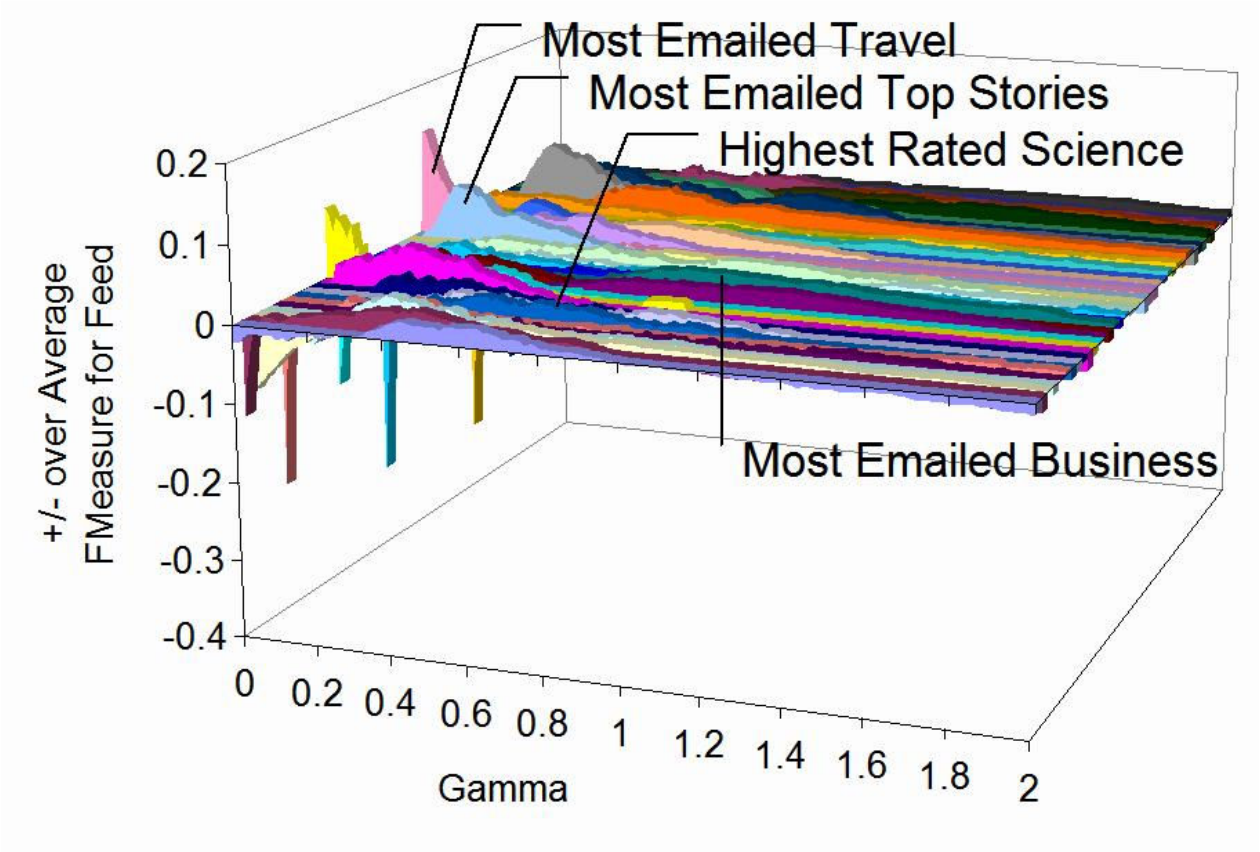

Figure 3.Each area curve is the normalized final F-Measure (y-axis) of each instantiation (x-axis). Curves for each interest-driven feeds from the Yahoo! News collection are shown (z-axis).

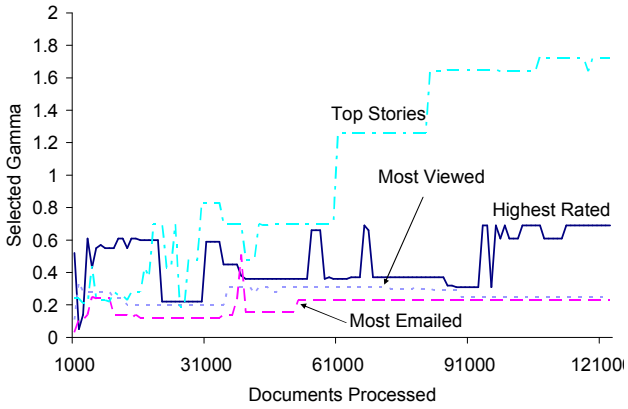

Figure 4. γ-values over time for a select number of feeds from the Yahoo! News collection.

articles and for negatively labeled articles, respectively. However, because γ is a weight relative to χ, we will evaluate multiple γ-values simultaneously while holding χ to 1. We call this scheme eRocchio.

Each document is evaluated by multiple instantiations of the Rocchio formulation in parallel, each with a different negative article weight γ, as shown in Figure 2. In our experiments, we evaluated all possible γ-values between 0 and 2, inclusive, in intervals of 0.01. Because the cosine similarity between the query profile and the document is a real number bounded between 0 and 1, and a binary decision must be made, the similarity is thresholded such that articles with a high similarity with the profile are labeled as interesting and articles with low similarity are labeled as uninteresting. Rather than use a static threshold, the efficacy of every threshold between 0 and 1 in increments of 0.01 is evaluated. Each Rocchio instantiation has its own adaptive thresholder to optimize its corresponding instantiation. Consequently, no particular distribution of interesting and uninteresting articles is assumed. And in the case of ties between utility measures, the threshold that yields the largest separation between interesting and uninteresting articles is used. Each instantiation of Rocchio has its own unique γ and adaptive

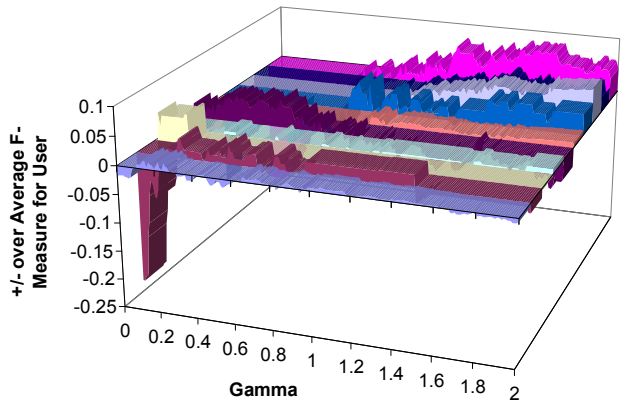

Figure 5. Each area curve is the normalized final F-Measure (y-axis) of each instantiation (x-axis). Curves for each of the users from the volunteer tagger collection are shown.

thresholder.

After each adaptive thresholder has generated a binary score from its corresponding Rocchio instantiation's generated similarity score, the evaluator must generate a final answer. The best Rocchio instantiation and its corresponding threshold are chosen by selecting the Rocchio instantiation and the threshold combination that has had the best utility measure up to that point. In our experiments, we use F_β-measure, which is the harmonic average of precision and recall, defined as:

$$F_\beta = \frac{(1+\beta)precision * recall}{\beta * precision + recall} \quad (6)$$

$$precision = \frac{|\, InterestingArticles \,\text{Re}\, trieved \,|}{|\, Articles \,\text{Re}\, trieved \,|} \quad (7)$$

$$recall = \frac{|\, InterestingArticles \,\text{Re}\, trieved \,|}{|\, InterestingArticles \,|} \quad (8)$$

For β, we use β=0.5, weighting precision twice as much as recall, which is consistent with the utility measure used in the TREC adaptive filter task [10].

In summary, a document is evaluated with the following steps:

1. A TF-IDF vector for the document is generated. Stop words are removed and the remaining terms are stemmed.
2. For each Rocchio instantiation, the cosine similarity of the document with the instantiations' stored profile (also a TF-IDF vector) is evaluated, using Equation 4.
3. For each Rocchio instantiation, the cosine similarity, computed in the previous step, is thresholded with the instantiation's currently best threshold, generating a binary score.
4. The binary score generated by the currently best instantiation is used as the final output of eRocchio.

After the actual interestingness of the document is revealed, eRocchio is updated as follows:

1. For each Rocchio instantiation, the profiles are updated using Equation 3.
2. The F_β-measure statistic for each instantiation is updated.
3. The adaptive threshold for each instantiation is updated by updating the F_β-measure statistic of every possible threshold for the instantiation.

It is expected that the computational cost for running eRocchio is proportional to the number of γ-values evaluated and the runtime of Rocchio. Thus, the runtime performance would be O(VR), where V is the number of γ-values evaluated and R is the runtime of Rocchio. Although, this runtime may seem large, with the availability of large-scale cluster computing, the multiple instantiations may be evaluated in parallel.

4. EXPERIMENTAL RESULTS

The eRocchio algorithm and iScore are implemented with the Apache UIMA framework [11]. The source code is available at [12].

We evaluate eRocchio and other classifiers from the machine

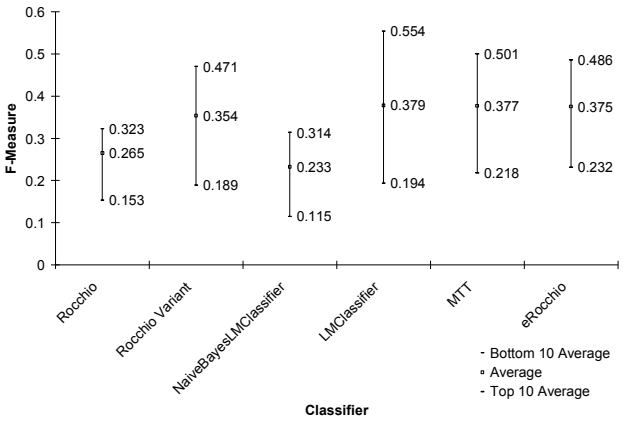

Figure 6. Average, low, and high F-Measure of different classifiers for the Yahoo! News collection.

learning community for two different tasks: recommending interesting articles and recommending relevant articles. Classifiers evaluate the interestingness/relevancy of articles, one at a time, in publication order. The classifiers are given no prior training data so they must learn as documents are streamed to the classifier. Only when the classifier makes a determination regarding the interestingness/relevancy of the article is the actual interestingness/relevancy revealed to the classifier. The classifier then is allowed to update itself with this new knowledge in preparation for the evaluation of the next article.

An *interesting* article is an article that an arbitrary user finds interesting. For interesting article recommendation task, two datasets were used. The first dataset is a set of 123,653 news articles from all Yahoo! News RSS feeds [13], collected over a span of one year. The interesting classification task is to identify the most interesting articles from this entire pool of articles for different communities of users. A community of users is determined by an interest-driven RSS feed from the Yahoo! articles collection. The 43 interest-driven RSS feeds considered for labeling are feeds of the form: "Top Stories [*category*]", "Most Viewed [*category*]", "Most Emailed [*category*]", and "Most Highly Rated [*category*]," including category-independent feeds such as the "Top Stories," "Most Emailed," "Most Viewed," and "Highest Rated" feeds. For example, RSS feeds such as

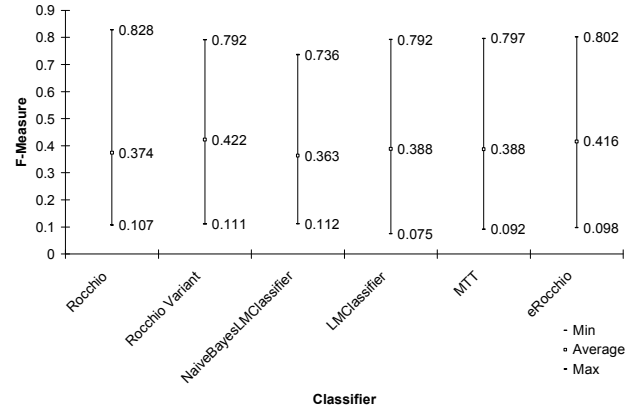

Figure 7. Average, low, and high F-Measure of different classifiers for the volunteer tagger collection.

"Most Viewed Technology" is a good proxy of what the most interesting articles are for technologists. Other categories, such as "Top Stories Politics," are a collection of news stories that the Yahoo! political news editors deem to be of interest to their audience, so the feed also would serve well as a proxy for interestingness. Note that these feeds are interest-driven and not category-driven, so the classification task is not the classical category classification task, but rather a more complex interest classification task. In the interest classification task, two articles that belong to the same topic may not necessarily of equal interest to a user or a community of users.

The second data set consists of articles collected from volunteer news readers that tag articles as they read their daily news on the web. A user can tag an article using a Firefox plugin or a Google News GreaseMonkey script add-on for Firefox. When a user tags an article as interesting or uninteresting, the plug-in or script records the webpage's URL and the user's tag as well as all the URLs contained within the referring webpage. Articles that are pointed by links from the referring webpage that have not been read by the user are considered as uninteresting for the user since the user deemed the title of the article to be uninteresting enough to not click on. The webpages are downloaded every night. Webpages that are non-articles (e.g., advertisements, table of contents, videos) are manually removed from the collection. We have 10 users that have read and tagged at least 50 articles. The entire document collection consists of 33,343 articles. A classifier is run for each user over only the documents that have been seen by a user as indicated by a user tagging or by existing on a referring page of a tagged article.

A *relevant* article is an article that is relevant to a specific query. We evaluate the relevancy classification task using the dataset and evaluation framework used in TREC11 [10]. The data set used for evaluating relevancy performance is the Reuters RCV1 corpus and a set of assessor manual taggings for 50 topics, such as "Economic Espionage." For example, a relevant article to the "Economic Espionage" query would be one that is related to economic espionage. The corpus is a collection of 723,432 news articles from 1996 to 1997. While the TREC adaptive filter task aims to evaluate algorithms that can return articles that are relevant to a query, not all articles that are relevant to a query are interesting. For example, articles that provide the same information may be relevant to a query, but are not necessarily interesting. 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.

4.1 User-dependent Variations for γ

Figures 3 and 5 show that that the choice of the optimal γ can be radically different, depending on the target feed/user. Each area curve is the normalized final F-Measure of each instantiation. The final F-Measure statistic has been normalized such that the graph shows the deviation from the average final F-Measure for a given feed/user.

Figure 3 shows the normalized F-Measure performance of each interest-driven feed from the Yahoo! News collection. Depending on the feed, the rate of deemphasizing uninteresting articles varies. For a feed such as "Most Emailed Travel," the best γ weight is near 0, meaning that uninteresting terms are forgotten very slowly. For "Most Emailed Top Stories" feed, the best γ weight is between 0.2 and 0.4 For the "Highest Rated Science"

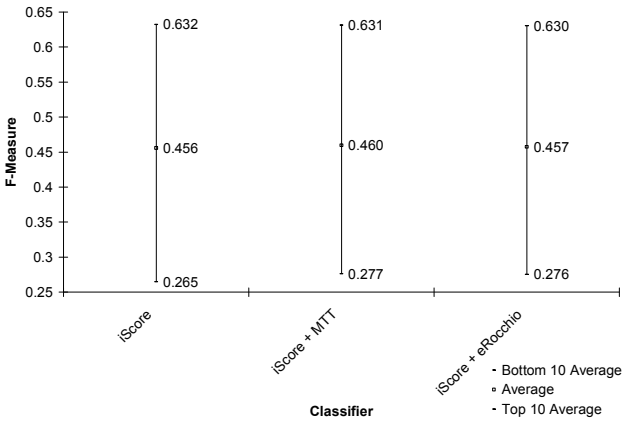

Figure 9. Average, low, and high F-measure of different iScore configurations for the Yahoo! News collection.

feed, the best γ weight is between 0.4 and 0.8. And for the "Most Emailed Business" feed, the best γ weight is much higher, between 1.0 and 1.4, meaning uninteresting terms are forgotten more quickly than the rate that interesting terms are reinforced.

Figure 4 shows the selected γ-values by eRocchio for the topic-independent interest-driven feeds from the Yahoo! News collection. For most of these feeds, eRocchio settles on a γ-value less than 1.0, except for "Top Stories," in which eRocchio selects a γ-value that seems to continually grow. This variation of γ-values over time is likely due to the behavior and type of news read by users represented by the feed. For example, users represented by the "Top Stories" feed may continually want to deemphasize terms from old uninteresting news very quickly; whereas for users represented by the "Most Viewed" feed do not want to deemphasize terms from old news as quickly.

Figure 5 shows the normalized F-measure performance of each user from the volunteer tagger collection. For half of the users, a low γ weight is optimal; whereas, for the other half of users, a high γ weight is ideal.

4.2 Recommendation Performance

Figure 6 shows the average F-Measure performance of several classifiers on the Yahoo! News collection, such as Rocchio, a Rocchio variant that was the best performing of in the last run of

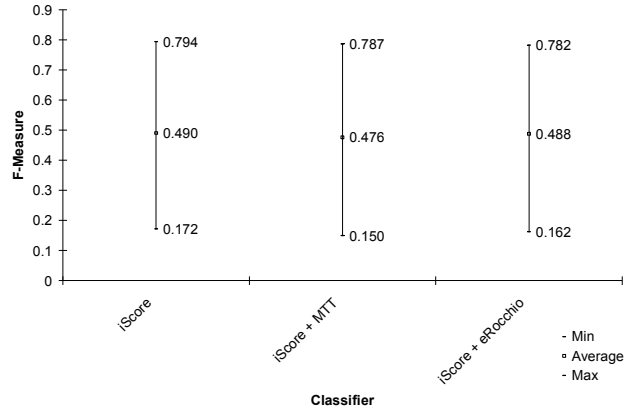

Figure 8. Average, low, and high F-measure of different iScore configurations for the volunteer tagger collection.

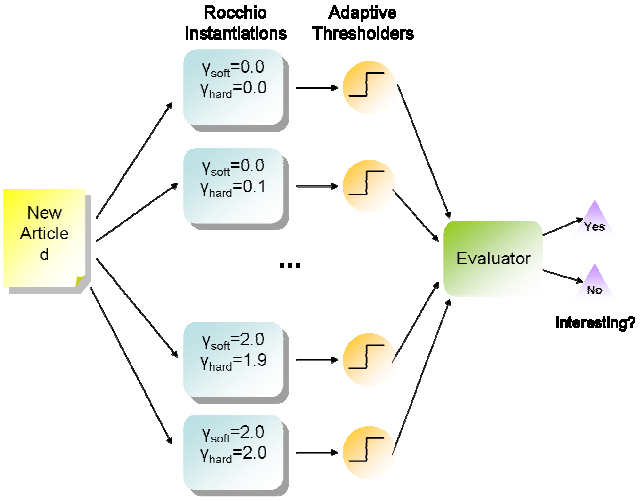

Figure 10. eRocchio pipeline with weights for soft and hard negatively-labeled articles.

the TREC Adaptive Filter Task [4], a naïve Bayesian language model classifier (NaiveBayesLMClassifier) [14], and a state-of-the-art language model classifier (LMClassifier) [14], multiple topic tracking (MTT) [7], and eRocchio. The figure also shows the average of the top 10 and bottom 10 performing feeds. The figure shows that eRocchio performs as well as the top classifiers, LMClassifier and MTT, despite its simpler algorithm, compared to LMClassifier, and the lack of parameter tuning, compared to MTT. It also shows that eRocchio outperforms the Rocchio variant by a significant 2.1 F-Measure points (6% improvement), indicating that online parameter selection can outperform a static *a priori* parameter selection. Although, eRocchio does not perform as well as the top classifiers with regards to the top 10 average, eRocchio performs better than all the other classifiers for the bottom 10 feeds, which are the most difficult to recommend articles for.

Figure 7 shows the average F-Measure performance of the same classifiers on the volunteer tagger collection. The figure also shows the worst and best performing users. In this dataset, in contrast to the previous dataset, LMClassifier and MTT do not perform as well as the Rocchio variant. However, eRocchio performs as well as the Rocchio variant, despite eRocchio's lack of parameter tuning that is required of the Rocchio variant.

Figure 8 shows several different iScore configurations evaluating the Yahoo! News collection. The basic iScore configuration (iScore) includes all the features detailed in [2], including Rocchio, the Rocchio variant, NaiveBayesLMClassifier, and LMClassifier. The iScore + MTT configuration includes MTT in addition to all features in iScore. The iScore + eRocchio configuration includes eRocchio in addition to all features in iScore. The figure shows that when MTT or eRocchio is added to the original iScore features, performance improves marginally, with the greatest improvement for the most difficult feeds. The figure also shows that when MTT is replaced with the much simpler eRocchio, performance remains relatively the same.

Figure 9 shows the same iScore configurations for the volunteer tagger dataset. Although replacement of MTT with eRocchio does improve recommendation results, it achieves as good of a performance as iScore with the original features with this dataset,

which is consistent with the results of Figure 7. Figure 7 shows that MTT does not perform as well as the Rocchio variant so it is expected that the inclusion of MTT would not improve recommendation results as indicated in Figure 9. However, eRocchio performs as well as the best of the base classifiers (i.e., the Rocchio Variant), so its inclusion into iScore yields similar performance. A larger dataset with more users may be necessary to make a more conclusive conclusion with regards to iScore and iScore + MTT.

4.3 TREC11 Retrieval Performance

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 eRocchio perform against other adaptive filters from TREC11. eRocchio is compared with the best filters from each participating group in TREC11 on the TREC11'S RCV1 corpus in Figure 11. In this set of experiments, eRocchio is adapted to learn from the initial training articles and the query description in an identical fashion to ICTAdaFT11Ub [4]. Also eRocchio is augmented to handle both soft and hard negative articles as shown in Figure 10. Consequently, instead of learning a parameter configuration consisting of only one parameter, eRocchio, in this set of experiments, learns a parameter configuration consisting of two parameters, one for hard negative articles and one for soft negative articles. The soft and hard negative article weights considered are between 0 and 2.0 in increments of 0.1. Each possible pair of soft and hard negative article weights are evaluated in parallel as the articles are processed one at a time.

Figure 11 shows that eRocchio outperforms the best classifier, ICTAdaFT11Ub, from the TREC11 run by a very significant 4.3 F-Measure points (10%. improvement). ICTAdaFT11Ub is the same algorithm as the Rocchio variant in the pervious experiments. This is a significant improvement in this area of work, where even small improvements are difficult to achieve. The improvement is due to the large increase in precision by eRocchio over ICTAdaFT11Ub, despite the slight drop in recall. eRocchio is similar to ICTAdaFT11Ub except that instead of using fixed static weights for negative articles across all query topics, eRocchio learns dynamically those parameters that are more suited to an individual query topic. The figure shows that the online learning of parameters specific for a query can also improve information retrieval results in addition to news article recommendations.

5. CONCLUSION

Future work will focus on studying eRocchio on a much larger volunteer tagger collection so that a more significant conclusion can be made between eRocchio and the Rocchio variant. Other work will also look at finding ways to efficiently search the parameter space for more complex algorithms, such as MTT and language model classifiers.

We have shown that optimal learning behavior for a classifier varies from user to user, so instead of using a fixed parameter configuration across all users, better recommendation results can be achieved by tailoring the parameters to a specific user. By evaluating the efficacy of several parameter configurations as documents are processed, a good parameter configuration can be determined in an online fashion, adapting to changes in the data

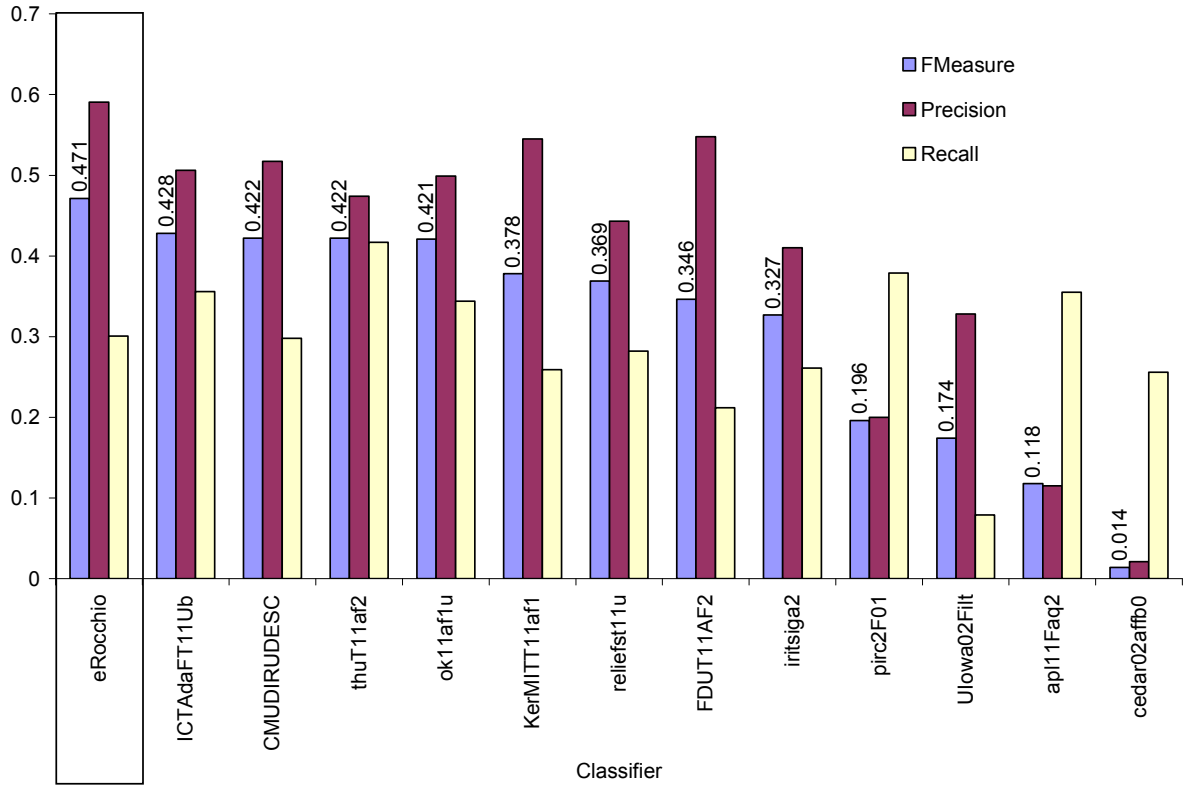

Figure 11. Performance in the TREC11 Adaptive Filter Task.

set and user behavior. Because of the effectiveness, simplicity, and adaptability of the eRocchio, it can replace algorithms such as Rocchio and MTT in iScore.

We have shown that online learning of parameter configurations can yield better news recommendation results from the Yahoo! news feeds. We have also shown that by adapting our algorithm, it can yield 10% better results than best performing filter from the TREC11 adaptive filter task. By learning parameters of a simple algorithm online for a specific user, similar recommendation performance can be achieved as more complex algorithms or algorithms that require extensively fine-tuning.

6. ACKNOWLEDGMENTS

This work (LLNL-CONF-405224) was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

7. REFERENCES

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

[2] 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.

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

[4] 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

[5] R. E. Schapire, Y. Singer, and A. Singhal, "Boosting and rocchio applied to text filtering," in *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval* (W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, eds.), (Melbourne, AU), pp. 215–223, ACM Press, New York, US, 1998.

[6] 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*, pp. 25–32, July 1997.

[7] R. K. Pon, A. F. Cardenas, D. Buttler, and T. Critchlow, "Tracking multiple topics for finding interesting articles," in *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, (New York, NY, USA), pp. 560–569, ACM Press, 2007.

[8] Z. Chen and B. Zhu, "Some formal analysis of Rocchio's similarity-based relevance feedback algorithm," *Inf. Retr.*, vol. 5, no. 1, pp. 61–86, 2002.

[9] T. Joachims, "A probabilistic analysis of the Rocchio algorithm with TF-IDF for text categorization," Tech. Rep. CMU-CS-96-118, Carnegie Mellon University, 1996.

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

[11] Apache, "Apache UIMA." [Online] http://incubator.apache.org/uima/, 2008.

[12] R.K. Pon, "iScore." [Online] http://sourceforge.net/projects/iscore/, 2008.

[13] Yahoo, "Yahoo news RSS feeds." [Online] http://news.yahoo.com/rss, 2008.

[14] Alias-I, "LingPipe," [Online]. http://www.alias-i.com/lingpipe/index.html, 2008.