Measuring the Interestingness of Articles in a Limited User Environment

R. K. Pon

October 14, 2008

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

Measuring the Interestingness of Articles in a Limited User Environment

by

Raymond K. Pon

2008

The dissertation of Raymond K. Pon is approved.

_____

David J. Buttler

_____

Jonathan Furner

_____

Junghoo Cho

_____

Wesley W. Chu

_____

Alfonoso F. Cárdenas, Committee Chair

University of California, Los Angeles

2008

To Popo

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my official and unofficial committee members, Alfonso Cardenas, Wesley Chu, John Cho, Jonathan Furner, Dave Buttler, and Terence Critchlow for their patience and guidance in completing my dissertation. Thank you, Professor Cardenas, for funding me and giving me research projects in my early days of graduate school and for your continuous encouragement and mentorship throughout the years. Thank you, Professors Chu and Cho, for inspiring me with your classes. Thank you, Professor Furner, for serving on my committee. Thank you, Terence, for giving me my first experience in a large-scale research environment and giving me a taste of what it means to be a researcher. Although, you could not serve on my committee in the end, you helped jump-start this whole research process. And major thanks to Dave, who has been my technical mentor and friend throughout this process, giving me some really odd-ball ideas (some of which paid off... and some that haven't) and even giving me a place to crash while I was in Livermore. I would also like to thank the LLNL staff, particularly Pam Mears, Linda Becker, Cindy Bottero, Jane MacNamara, Jim McGraw, Joanna Allen, Maya Gokhale, and the late Marcus Miller. The resources and funding at LLNL were invaluable in completing my experiments and my conference travels. The UCLA staff has also been helpful in my career at UCLA, particularly Terry Valai and Verra Morgan. There are also some other people that have been helpful in the writing of this dissertation. The figure regarding the iScore implementation pipeline was drawn by Bassam Islam. The tagger dataset was made possible by the gracious anonymous volunteers that have tagged articles for me.

I would also like to thank my friends and family for their support. It has been a rough few years, but all of you were always there. Thank you, Mom and Dad, for supporting my education and giving me a home throughout my life. I

have grown to appreciate the sacrifices you have made so that I could have a better life. Thank you, Sharon, Kelly, and James, for being my spiritual guides during this critical point in my life. My Sedaqah group has also been there for my spiritual and moral support. Your prayers have been very much appreciated, especially when I became homesick during my tours of duty at LLNL. Thank you, Vince, Jeff, and Rex, for sticking around the UCLA area; otherwise, I wouldn't have anyone to hang around with. Thank you, Rod, for being my unofficial PhD mentor and for helping me navigate the waters at UCLA. And many thanks and love to my girlfriend, Joyce, who has always believed in me even when I didn't.

And I would like to finally thank God, from whom all good things come, particularly this body of work. I am very thankful that He led me to Him during this crazy period of my life.

1998   Valedictorian, San Gabriel High School, California.

2000–2001   Software Engineer Summer Intern, Advance Design System, Agilent Technologies, Westlake Village, California. Designed and implemented new layout tools and other user-interfaces.

2001   Undergraduate Student Researcher, Multimedia Information Stream System Technology Group, UC Los Angeles, California.

2001   B.S. (Computer Science and Engineering), UC Los Angeles, California.

2002–2005   Graduate Student Researcher, Multimedia Information Stream System Technology Group, UC Los Angeles, California.

2003   M.S. (Computer Science), UC Los Angeles, California.

2004–2005   Technical Scholar, Computational Directorate, Lawrence Livermore National Laboratory, California.

2007   Teaching Assistant, Computer Science Department, UC Los Angeles, California. Taught Computer Science 35L: Software Construction Laboratory under the direction of Professor Paul Eggert.

2006–Present   Lawrence Scholar, Computational Directorate, Lawrence Livermore National Laboratory, California.

Publications and Presentations

R.K. Pon, A.F. Cárdenas, and D. Buttler, "Online Selection of Parameters in the Rocchio Algorithm for Identifying Interesting News Articles," presented at 10th ACM International Workshop on Web Information and Data Management (WIDM), Napa Valley, CA, October 30, 2008.

R.K. Pon, "Measuring the Interestingness of Articles in a Limited User Environment," presented at Yahoo! News, Sunnyvale, CA, September 24, 2008.

R.K. Pon and D. Buttler, "Metadata Registry, ISO/IEC 11179," Encyclopedia of Database Systems, 2008, in press.

R.K. Pon, A.F. Cárdenas, and D. Buttler, "Improving Naïve Bayes with Online Feature Selection for Quick Adaptation to Evolving Feature Usefulness," presented at 2008 SIAM SDM Text Mining Workshop, Atlanta, GA, April 26, 2008.

R.K. Pon, A.F. Cárdenas, and D. Buttler, "Measuring the Interestingness of Articles," in J. Wang (ed), Encyclopedia of Data Warehousing and Mining (Second Edition), IGI Global, 2008, in press.

R.K. Pon, "Measuring the Interestingness of Articles in a Limited User Environment," presented at Yahoo! Research, Burbank, CA, February 10, 2008.

R.K. Pon, A.F. Cárdenas, D. Buttler, and T. Critchlow, "Tracking Multiple Topics

for Finding Interesting Articles," presented at 2007 SIGKDD, San Jose, CA, August 12-15, 2007.

R.K. Pon, A.F. Cárdenas, D. Buttler, and T. Critchlow, "iScore: Measuring the Interestingness of Articles in a Limited User Environment," presented at IEEE Symposium on Computational Intelligence and Data Mining 2007, Honolulu, HI, April 1-5, 2007.

D.A. Aoyama, J.T. Hsiao, A.F. Cárdenas, and R.K. Pon, "Timeline and Visualization of Multiple Datasets and the Visualization Querying Challenge," Journal of Visual Language and Computing, vol. 18, no. 1, 2006.

S.E. Chan, R.K. Pon, and A.F. Cárdenas, "Visualization and Clustering of Author Social Networks," presented at 2006 International Conference on Distributed Multimedia Systems (DMS 2006) Workshop on Visual Languages and Computing, Grand Canyon, AZ, August 30-31, 2006.

Q. Zhou and R.K. Pon, "A Relational Solution to DNA Sequence Matching," presented at 15th International Conference on Software Engineering and Data Engineering (SEDE 2006), Los Angeles, CA, July 6-8, 2006.

A. F. Cárdenas, R. K. Pon, and B. S. Islam, "The Image Stack Stream Model, Querying, and Architecture," presented at 2005 International Conference on Distributed Multimedia Systems (DMS 2005), Banff, Canada, September 5-7, 2005.

R. K. Pon and T. Critchlow, "Performance-Oriented Privacy-Preserving Data

Integration," presented at Data Integration in the Life Sciences 2005, San Diego, CA, July 20-22, 2005.

R. K. Pon and A. F. Cárdenas, "Data Quality Inference," presented at Second International ACM SIGMOD Workshop on Information Quality in Information Systems (IQIS 2005), Baltimore, MD, June 17, 2005.

A.F. Cárdenas, R.K. Pon, R.B. Cameron, and M.A. Coyle, "The Mobile Patient and the Mobile Physician Data Access and Transmission," in the Proceedings of the 2005 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS '05), Las Vegas, NV, June 20-23, 2005.

A.F. Cárdenas, R.K. Pon, P.A. Michael, and J.T. Hsiao, "Image Stack Viewing and Access," Journal of Visual Language and Computing, vol. 14, no. 5, pp.421-441, 2003.

A.F. Cárdenas, R.K. Pon, and R.B. Cameron, "Management of Streaming Body Sensor Data for Medical Information Systems," in the Proceedings of the 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS '03), Las Vegas, NV, June 23-26, 2003.

Abstract of the Dissertation

Measuring the Interestingness of Articles in a Limited User Environment

by

Raymond K. Pon

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2008

Professor Alfonoso F. Cárdenas, Chair

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 varies from user to user. Although methods such as collaborative filtering have been shown to be effective in recommendation systems, in a limited user environment, there are not enough users that would make collaborative filtering effective.

A general framework, called iScore, is presented for defining and measuring the "interestingness" of articles, incorporating user-feedback. iScore addresses various aspects of what makes an article interesting, such as topic relevancy, uniqueness, freshness, source reputation, and writing style. It employs various methods to measure these features and uses a classifier operating on these features to recommend articles. The basic iScore configuration is shown to improve recommendation results by as much as 20%. In addition to the basic iScore features, additional features are presented to address the deficiencies of existing

feature extractors, such as one that tracks multiple topics, called MTT, and a version of the Rocchio algorithm that learns its parameters online as it processes documents, called eRocchio. The inclusion of both MTT and eRocchio into iScore is shown to improve iScore recommendation results by as much as 3.1% and 5.6%, respectively. Additionally, in TREC11 Adaptive Filter Task, eRocchio is shown to be 10% better than the best filter in the last run of the task.

In addition to these two major topic relevancy measures, other features are also introduced that employ language models, phrases, clustering, and changes in topics to improve recommendation results. These additional features are shown to improve recommendation results by iScore by up to 14%. Due to varying reasons that users hold regarding why an article is interesting, an online feature selection method in naïve Bayes is also introduced. Online feature selection can improve recommendation results in iScore by up to 18.9%.

In summary, iScore in its best configuration can outperform traditional IR techniques by as much as 50.7%. iScore and its components are evaluated in the news recommendation task using three datasets from Yahoo! News, actual users, and Digg. iScore and its components are also evaluated in the TREC Adaptive Filter task using the Reuters RCV1 corpus.

CHAPTER 1

Introduction

An explosive growth of online news has taken place in the last few years. Users are inundated with thousands of news articles, only some of which are interesting to them. A system to filter out uninteresting articles would aid users that need to read and analyze many news articles daily, such as financial analysts, government officials, and news reporters. Information overload is a threat to a user's ability to function, resulting in "brain-thrashing" [Den06], calling for a VIRT (valued information at the right time) [Hay06] strategy for information handling.

The most obvious approach for a VIRT strategy is to learn keywords of interest for a user [CCG04, LLK03, BPC00]. Unfortunately, the issues related to article recommendation systems are more difficult to address than applying a simple keyword filter to weed out uninteresting articles. Although filtering articles based on keywords removes many irrelevant articles, there are still many uninteresting articles that are highly relevant to keyword searches. For example, searching for "San Francisco" in Google News will yield about 135,000 articles ordered by relevance. Unfortunately, a relevant article may not be interesting for various reasons, such as the article's age or if it discusses an event that the user has already read about in other articles.

Although it has been shown that collaborative filtering can aid in personalized recommendation systems [WVR06] a large number of users is needed. In a limited user environment, such as a small group of analysts monitoring news events,

collaborative filtering would be ineffective. To address this insufficiency to news filtering, a different approach is taken by 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 the presented system, iScore [PCB07a], the following contributions are made to news filtering in a limited user environment including the development of a prototype system:

1. Filtering based on only topic relevancy is shown to be insufficient for identifying interesting articles.

2. A variety of features are extracted, 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 20% higher quality results. For each user, these features have different degrees of usefulness for predicting interestingness.

3. Several classifiers are evaluated 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, is shown to not capture all aspects of personalized news filtering systems necessary for system evaluation.

Despite incorporating other article features in addition to relevancy to topics of interest, the initial version of iScore still performs poorly with users that have broader interests (as opposed to 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. The initial version of iScore is extended 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) [NIS04] 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, the following additional contributions through multiple topic tracking (MTT) [PCB07b] are made:

1. Instead of identifying all new topics and tracking all articles for those topics as in TDT, MTT focuses on the specific user's interests, which are constantly evolving. Focusing on only evolving user interests instead of all topics allows for more efficient resource utilization.

2. The use of multiple profile vectors yields better results than traditional methods, such as the Rocchio algorithm, for identifying interesting articles. Additionally, multiple topic tracking as a new feature in iScore improves

iScore classification performance.

3. For a specific user as a case study, the operating parameters for the MTT algorithm are analyzed for their classification performance.

However, methods like MTT, require extensive parameter tuning. How parameters can be learned online for an algorithm, like Rocchio, is also studied. In this area, the following contributions are made:

1. Users are shown to have different learning/reading behavior when evaluating the interestingness of news articles.

2. Instead of using static parameters, several different parameter configurations are evaluated simultaneously for a simple IR algorithm so that similar or better recommendation results can be achieved, compared to more complex information retrieval algorithms (e.g., language modeling classifiers) and algorithms that require fine-tunining (e.g., MTT and Rocchio).

3. By tailoring an algorithm specifically to an user instead of using an "one-size-fits-all" algorithm, better recommendation results can be achieved.

Furthermore, the definition of what makes an article interesting varies from user to user and continually evolves, even for a single user. As a result, for news recommendation systems, useless document features can not be determined *a priori* and all features are usually considered for interestingness classification. Consequently, the presence of currently useless features degrades classification performance [For04], particularly over the initial set of news articles being classified. The initial set of documents is critical for a user when considering which particular news recommendation system to adopt. To address these problems, an improved version of the naïve Bayes classifier is introduced with online feature

selection [PCB08a]. Correlation is used to determine the utility of each feature and leverage the conditional independence assumption used by naïve Bayes for online feature selection and classification. The following contributions are made in this area:

1. Augmenting naïve Bayes with online feature selection allows for the most proper features to be used, improving iScore's performance.

2. The continual learning of statistics about each feature allows for the invocation of any feature at any time if it has been determined to be useful, addressing the problem of the evolving definition of interestingness.

3. By only considering the top-k useful features, evaluation of all possible subsets of features is avoided, making the presented feature selection approach tractable in an online setting.

This dissertation is organized as follows. Chapter 2 discusses the works related to news recommendation based on content analysis. Chapter 3 illustrates the problem that this dissertation addresses and how it is modeled in experimental evaluations with the various datasets available. Chapter 4 discusses the basic iScore framework that is used to address the recommendation problem discussed. Chapter 5 introduces an additional topic relevancy feature that tracks multiple topics. Chapter 6 addresses the problem that many topic relevancy features have in selecting the appropriate parameters and introduces online parameter selection in the Rocchio algorithm. Chapter 7 introduces additional features that have shown to improve recommendation performance. Chapter 8 addresses the problem of users having differing reasons as to why an article is interesting and introduces online feature selection in naïve Bayes. Chapter 9 summarizes the findings of this dissertation from experimental results. Chapter 10 explores other

roads of research that may be pursued to extend iScore to improve its usability and effectiveness. The Appendix includes a discussion on the implementation of iScore in its experimental setup, other roads of research considered that did not yield fruit, reference tables, and statistical significance test results.

CHAPTER 2

Related Works

2.1 Recommendation systems

iScore is a recommendation system in a limited user environment, so the only available information is the article's content and its metadata. Because of the limited user environment, methods that require many multiple users, such as collaborative filter, are not effective. Work outside collaborative filtering makes use of this information in a variety of ways. Work by [CGR05] ranks news articles and new sources based on several properties in an online method. They claim that important news articles are clustered based on topics. 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. However, mutual reinforcement calculations require multiple passes over a document collection and are not possible in a single-pass online environment, such as in iScore. In iScore, news articles are ranked based on various properties in an online method, but instead of ranking articles using mutual reinforcement and article freshness, a different variety of features is studied. Additionally, when training the classifiers in iScore, there is a slightly different definition of freshness. For [CGR05], the notion of freshness indicates that a recently written article may be more interesting than older articles. The definition of iScore is slightly different, where topics that have interesting articles recently published may be

more interesting.

Another approach taken by [MP01] 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 term-frequency-inverse-document frequency (TF-IDF) weights for tokens. A near neighbor text classifier [BPC00] uses a document vector space model. A personalized multi-document summarization and recommendation system by [RFZ01] recommends articles by suggesting articles from the same clusters in which the past interesting articles are located. In iScore, a variation of these methods are implemented as one of many feature extractors in iScore. Another clustering approach, MiTAP [DWK03] 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 since this approach bins but does not recommend articles.

Although it has been shown that collaborative filtering can aid in personalized recommendation systems [WVR06] 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, a different approach is taken in iScore by identifying what makes an article interesting.

2.2 Adaptive filtering

The work in iScore is closely related to the adaptive filtering task in the Text Retrieval Conference (TREC), which is the online identification of news articles that are most relevant to a set of topics. The task is different from identifying interesting articles for a user because an article that is relevant to a topic may not necessarily be interesting. However, relevancy to a set of topics of interest is an indicator of for interestingness. The report by [RS02] summarizes the results of the most recent 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, iScore uses adaptive thresholds and incremental profile updates.

In [XYW02], 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, this method is incorporated into iScore. Other methods explored in TREC11 include using a second-order perceptron, a support vector machine (SVM), a Winnow classifier [WHN02], language modeling [MCM02], probabilistic models of terms and relevancy [Bro02], and the Okapi Basic Search System [RWZ02].

2.3 Ensembles

Other work, like iScore, have leveraged multiple existing techniques to build better systems for specific tasks. For example, in [Hen06], the authors combine two popular webpage duplication identification methods to achieve better results.

Another example is by [LK05], which combines the results from multiple outlier detection algorithms that are applied using different sets of features. Other examples for combining multiple techniques is in the Netflix Prize contest [Net07], where the team "When Gravity and Dinosaurs Unite" combine two different recommendation algorithms to achieve better results.

Other work in bagging and boosting identifies conditions bagging and boosting will outperform single classifiers and how to maximize the diversity of classifiers in the ensemble. In [ES04], Monte Carlo analysis is used to characterize the conditions which the ensemble approach will outperform the single classifiers. They provided a closed form expression for the distribution of ensemble accuracy, mean, and variance. In [MM04], the authors also address diversity in ensembles. Classifiers that are trained on the original data and some artificial data generated from a random process that follows the training data distribution are added to the ensemble if it does not increase the ensemble training error. In [TG04], a greedy algorithm is used for selecting models in ensembles. In the greedy framework, when a new instance is to be classified, a meta-model invokes the appropriate predictive model that best corresponds to the instance. They introduce a new algorithm for learning the base models and the meta-learner for model selection, which relies on moving small amounts of data between the various datasets that are used to train the base models. Data is moved according to a simulated annealing algorithm.

A closely related ensemble work by [YH06] 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. In the iScore experiments, two methods of a linear combination approach are explored, using correlation and logistic regression. But in contrast to [YH06], functions are combined that can not necessarily be used for ranking documents for interestingness by themselves. Each function is a different aspect of interestingness, and the functions need to be combined together to generate meaningful scores for interestingness.

2.4 Topic detection and tracking

Topic detection and tracking (TDT) identifies new events and groups news articles that discuss the same event. Formally, TDT consist of five separate tasks: (1) topic tracking, (2) first story detection, (3) topic detection, (4) topic linkage, and (5) story segmentation [NIS04].

Many TDT systems, like [APL98], [FWM01], and [All02] are simply a modification of a single pass clustering algorithm. They compare a news story against a set of profile vectors stored in memory. If the story does not match any of the profiles by exceeding a similarity threshold, the story is flagged as a new event and a new profile is created using the document vector of the news story. Otherwise, the news story is used to update the existing profiles. Other work, such as [MAS04], add simple semantics of locations, names, and temporal information to the traditional term frequency vectors used in previous work.

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

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

Another closely related work is in the discovery of evolutionary theme patterns (ETP) from text [MZ05]. In ETP, documents are partitioned into possibly overlapping subcollections according to their publication time. The most prominent themes (or subtopics) are extracted from each subcollection. For themes in two different subcollections, an ETP solution decides whether there is an evolutionary transition from one theme to the other. The general ETP problem is not restricted to operation within an online and continuous environment, so the solution posed by [MZ05] 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 [MZ05] does not learn which themes or topics are of interest to the user, and so all themes are maintained and are not useful for classifying the interestingness of an article. In ETP, the evolutionary relationships among themes at different times are also explicitly identified, which is not necessary for discovering the most interesting articles for the user as they are published.

2.5 Trust and quality

Articles known to be credible may be more interesting than articles that are not. Frequent reporting errors (or subjective disagreement) by a news agency may

contribute to a user deeming a future article uninteresting even before he reads it. To further the understanding of credibility assessment, in [Fog03], the authors present the prominence-interpretation theory in which two things happen when people assess credibility online: (1) the user notices something, and (2) the user makes a judgment about it. In [NSK06], the authors rate the credibility of news documents on the web with three metrics: commonality, numerical agreement, and objectivity. They hypothesize that as more news publishes deliver articles with similar content to the target article being assessed, the higher the credibility of the target article. Also, if numerical expressions that contradict those in other articles from different news agencies, credibility is rated lower. The credibility of the article containing subjective speculation is rated differently from those containing objective news sources. They use a dictionary approach and other heuristics to rate objectivity.

In [PC05], it is claimed that the quality of data sources can be estimated by observing the agreement among data sources and their past history of being correct. Similarly, [RE06] uses the value of information for evidence detection. They measure the reliability of data sources, the coherence (the agreement) among sources, and the independence between sources.

Trust and quality are addressed in iScore by examining the objectivity of articles and the reputation of the news agency for producing interesting articles. Objectivity is measured using a language model trained to differentiate between objective and subjective sentences. Source reputation is measured by simply looking at the proportion of interesting articles written by a particular author. However, addressing agreement among text documents is a difficult task and is an open area of research, which is beyond the scope of this dissertation.

13

2.6 Feature selection

Since the results of iScore's weak classifiers and the results of other feature extraction methods are used to build a large overall classifier, feature selection is important. But because the importance of features vary among users, is unknown *a priori*, and may change over time, no features can be discarded when constructing the overall classifier. For example, for a period of time, the writing style of an article may be important, but for a later period, it may not be as important as another feature, such as topic relevancy.

The work by [GE03] is a survey on feature selection, noting cases where feature selection would improve the results of classifiers. Noise reduction and better class separation may be obtained by adding features that are presumably redundant. Features that are independently and identically distributed are not truly redundant. Perfectly correlated features are truly redundant in the sense that no additional information is gained by adding them. However, very high feature correlation does not mean absence of feature complementarity. A feature that is completely useless by itself can provide a significant performance improvement when taken with others; consequently, two features that are useless by themselves can be useful together.

There have been three directions to feature selection: wrappers, filters, and embedded methods. Wrappers use the learning machine of interest as a black box to score subsets of features according to their predictive power. An example of a wrapper approach is [KJ97], which uses a hill-climbing approach to find a good set of features. Filters select subsets of features as a pre-preprocessing step, independently of the chosen predictor. Embedded methods perform feature selection and training, and are usually specific to given learning machines. To create a few baselines performance values to compare with when selecting features

for a new problem, [GE03] recommend a linear predictor (e.g., linear SVM) and select features in one of two ways: (1) a feature ranking method using correlation coefficient or mutual information; and (2) with a nested subset selection method performing forward or backward selection or with multiplicative updates.

There have been several other surveys on feature selection. The paper by [LY05] calls for the integration of different feature selection algorithms. They combine the filter and the wrapper approach into a single hybrid approach for feature selection. The hybrid approach uses the independent measure of the filter approach to decide the best subsets for a given cardinality and uses the mining algorithm of the wrapper approach to select the final best subset among the best subsets across different cardinalities. Another paper by [LDD05] discusses feature selection applied to real-life problems. The work by [BL97] discusses feature-weighting methods such as Winnow [Lit88]. The Winnow algorithm is very similar to that of the perceptron, except instead of additive updates, it uses multiplicative updates. Furthermore, the inputs and outputs of the Winnow algorithm are all binary. They have shown that the number of mistakes grows only logorithmically with the number of irrelevant attributes in the examples while still being computationally efficient in both time and space. [YP97] is a study on feature selection methods in statistical learning of text categorization. They evaluated five methods: term selection based on document frequency, information gain, mutual information, chi-square test, and term strength. They found that document frequency, information gain, and the chi-square test were most effective in their experiments. Their experiments suggest that document frequency thresholding, the simplest and lowest cost method, can be used reliably instead of the other two preferred methods.

Since decision trees use information gain to rank features, it can be considered

as an embedded feature selection method. Work by [UBC97] discusses an incremental decision tree algorithm that makes use of an efficient tree restructuring algorithm. However, the drawback is that any numeric data must be stored and maintained in sorted order by value and the decision tree's storage requirements will continually grow.

Other work in online feature selection addresses a different problem. In [PT03], techniques are studied for selecting features from a set of features that grow over time. Instead of a fixed set of features and a growing number of training instances to work from, the set of features continues to grow as the number of training instances remains fixed. However, in the iScore framework, the set of features with varying degrees of utility is fixed while the number of training instances continues to grow.

Another method for feature selection is to reduce the number of redundant features, which is different from our goal of reducing the number of irrelevant features. In [NF05], redundant features are identified by performing pair-wise similarities measurements using the properties of time series data, which may not be directly applied to news articles. In our experiments, we assume a more general setup, where documents from different news sources that span multiple domains are aggregated together into a single document stream and are simply ordered by publication time. Consequently, an article in the document stream is not necessarily dependent upon the content of the article that immediately precedes it in the document stream.

The feature selection method used by [XJK01] employs information gain ranking and Markov blanket filtering. Although, we rank features based on their correlation to the interestingness class similar to how Xing ranks features based on their information gain, we use correlation rather than information gain due to

correlation's computability in an online environment. Information gain requires the discretization of feature values which requires examining the entire range of possible values for a feature which is not possible in an online setting. The Markov blanket filtering is a more computationally intensive subset selection procedure, which is not ideal for an online setting either.

This survey on feature selection has shown that existing feature selection methods, wrappers and filters are not applicable in the online news recommendation problem. Inspired by the effectiveness of the embedded approach to feature selection, a new classifier is constructed in iScore that is a linear combination of features, which is determined dynamically by the correlation of their respective features and interestingness. Logistic regression is also studied as a method for embedded feature selection as well. After these initial experiments with feature selection, a method in the iScore framework is developed to allow online feature selection within naïve Bayes by examining multiple feature subsets simultaneously during classification.

2.7   Document classification

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

There has been a lot of work in document classification in recent years. In [WKY96], the authors classify documents, taking into account the term frequencies as well as the local relationships between available classes. They try to balance specificity, which measures the degree of precision with which the con-

tents of a document is represented by the classification result, and exhaustivity, which measures the degree of coverage by the classification result on the domain found in a document. In [Lia04], a SVM was used for web-page classification. In [AU06], distributional clustering and a logic-based learning algorithm are used to classify documents. In [AW06], the authors combine the graph/network properties of documents along with traditional content classification methods to classify documents. Work by [DM06] improves classification of documents by using multiple large external corpora. This is accomplished through a mixture of relevance models.

Latent Dirichlet Allocation (LDA), first proposed by [BNJ03] for document classification, has been popular in document classification. LDA is a generative probabilistic model for collections of discrete data such as text. It is a three-level hierarchical Bayesian model, in which each item of a collection is modeled as a finite mixture over an underlying set of topics. Each topic is modeled as an infinite mixture over an underlying set of topic probabilities. The topic probabilities represent a document. Works by [NCS06, GS04, Ste06] extend [BNJ03] using LDA. They use statistical topic models to classify documents into topics not known a priori, similar to unsupervised learning methods such as traditional clustering methods. Unlike most clustering methods, the clusters of documents may share documents (i.e., overlap). In their topic model, a topic is a multinomial probability distribution over unique words in the vocabulary of the corpus. Each document is a mixture of topics and is represented as a multinomial probability vector, one probability for each topic. Given this model for a set of documents and topics, Gibbs sampling is used to estimate the topic-word and document-topic distributions.

There has also been work on boosting the performance of existing document

classifiers. In [SA06], the authors evaluate the effectiveness of using similarity browsing as a tool, like relevance feedback, for improving retrieval performance. They achieve performance that matched that of a traditional styled iterative relevance feedback technique.

Work by [YZH03] classifies documents from positive and unlabelled documents; whereas, most classification schemes assume that the training data are completely labeled. They extend a support vector machine (SVM) for this task. They show that when the positive training data is not too under-sampled, their approach outperforms other methods because it exploits the natural gap between positive and negative documents in feature space. This situation is a scenario in which iScore mostly operates in, where most articles are unlabelled with a few documents that are positively labeled. Unfortunately, an SVM method is not applicable to iScore's online operating environment because most SVM methods are costly to update continuously.

Since the use of logistic regression for iScore is studied, it is important to note other uses of logistic regression in classification. In [GLM04], the authors use logistic regression for text categorization to eliminate the need for ad hoc feature selection. They solve their logistic regression problem using a variation of the Gauss-Sidel method. In [LHF05], the authors combine tree induction with logistic regression, where the trees' leaves contain linear regression functions.

CHAPTER 3

Modeling Users

3.1   User model

The news recommendation user model used in this dissertation follows the evaluation model in the TREC11 adaptive filter task [RS02]. 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.

3.2   Datasets

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

used: Yahoo! News, tagger and Digg. In addition to the recommendation task, the TREC Adaptive Filter task is also used to model a different type of recommendation task for some of the experiments in this dissertation. The datasets are summarized in Table 3.1.

3.2.1 Yahoo! News

The first dataset is a set of 35,256 and 123,653 news articles from all Yahoo! News RSS feeds [Yah07], collected over a span of three months and one year, respectively. The smaller dataset is used in a few of the earlier experiments of iScore; whereas, the larger dataset is used for the later evaluations of iScore. The "interestingness" classification task for the Yahoo! News dataset is to identify the most interesting articles from this entire pool of articles for different communities of users. A community of users is determined by an interest-driven RSS feed from the Yahoo! News articles collection. The 43 interest-driven RSS feeds considered for labeling are feeds of the form: "Top Stories [category]," "Most Viewed [category]," "Most Emailed [category]," and "Most Highly Rated [category]," including category-independent feeds such as the "Top Stories," "Most Emailed," "Most Viewed," and "Highest Rated" feeds. The list of RSS Feeds considered as users are detailed in Appendix C in Table C.4. For example, RSS feeds such as "Most Viewed Technology" is a good proxy of what the most interesting articles are for technologists. Other categories, such as "Top Stories Politics, are a collection of news stories that the Yahoo! political news editors deem to be of interest to their audience, so the feed also would serve well as a proxy for interestingness. Note that these feeds are interest-driven and not category-driven, so the classification task is not the classical category classification task, but rather a more complex classification task. In the "interestingness" classification task, two

articles that belong to the same topic may not necessarily be of equal interest to a user or a community of users.

3.2.2 Tagger

In the Yahoo! News dataset, the user is modeled after a community instead of as an individual interested in a particular category. Consequently, the second dataset consists of articles that are anonymously collected from volunteer news readers that tag articles as they read their daily news on the web. A user can tag an article using a Firefox plug-in or a Google Reader GreaseMonkey script add-on for Firefox. The script and plug-in do not record any personal identifiable information but instead uses a unique randomly generated identifier to uniquely identify users. When a user tags an article as interesting or uninteresting, the plug-in or script records the webpage's URL, the user's tagging, and the URLs contained within the referring webpage. Articles that are pointed by links from the referring webpage that have not been read by the user are considered as uninteresting for the user since the user deemed the title of the article not to be interesting enough to click on and tag. The webpages are downloaded every night. Webpages that are non-articles (e.g., advertisements, table of contents, videos) are manually removed by this author from the collection. The dataset consists of only 16 users that have read and tagged at least 50 articles. The entire document collection consists of 35,656 articles. A classifier is run for each user over only the documents that have been seen by a user as indicated by a user tagging or by existing on a referring page of a tagged article.

3.2.3   Digg

Due to the difficulties in recruiting to consistently read and tag articles, an alternative method is looked at, which is collected via the web service, Digg, to complement the user tagging collection. There have been 18,924 pages from the front page of Digg collected and "dugg" [Dig07] by the top 100 active Digg users [Fin08]. Of these 100 active Digg users, 63 have "dugg" more than 100 articles from the front page, which are considered as the set of users for this dataset. The usernames are listed in Appendix C in Table C.3. With this dataset, the task is to identify which articles have been "dugg" by a user from the "Popular Stories" section of Digg. It is assumed that the most active users will be aware of the majority of the articles on the "Popular Stories" section of Digg. Although this actual user dataset accurately reflects individual users, the dataset is much smaller and much noisier due to the heterogeneity of the type of pages being collected. Many of the webpages downloaded may not be news but images, videos, and other webpages beyond the scope of this dissertation.

3.2.4   TREC Adaptive Filter

Users are also modeled with a relevancy-based dataset. A relevant article is an article that addresses a specific query. The relevancy classification task is evaluated using the dataset and evaluation framework used in TREC11 [RS02]. The dataset used for evaluating relevancy performance is 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. In other words, a user is represented by a query. 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.

Collection	Size	Advantage/ Disadvantage
Yahoo! News: News articles collected from 43 feeds such as Top Stories Politics or Most Emailed Technology.	35,256 (small version) and 123,653 (large version)	**Advantage**: Large corpus, easily collected, and very clean data. **Disadvantage**: Can only evaluate iScore for users modeled as a collective community.
Tagger: Articles tagged by 16 volunteers along with the articles contained within the referring page of tagged articles.	45,656	**Advantage**: Evaluates iScore on individual users. **Disadvantage**: Difficult to collect and very small corpus.
Digg: Articles "dugg" by 63 of the most active Digg users.	18,924	**Advantage**: Easily collected, evaluates iScore on individual users. **Disadvantage**: Small corpus and very noisy data.
TREC11 Adaptive Filter: Reuters RCV1 Corpus.	723,432	**Advantage**: Very large corpus. **Disadvantage**: Can only evaluate iScore on topic relevancy.

Table 3.1: News collections

3.3 Evaluation metrics

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

$$precision = \frac{|\text{Int Articles Retrieved}|}{|\text{Articles Retrieved}|} \tag{3.1}$$

$$recall = \frac{|\text{Int Articles Retrieved}|}{|\text{Int Articles}|} \tag{3.2}$$

$$f_\beta = \frac{1 + \frac{\beta}{2}|\text{Articles Retr}|}{|\text{ Articles Retr}| + \frac{\beta}{2}|\text{Int Articles}|} \tag{3.3}$$

In this dissertation, FMeasure is the primary metric. FMeasure encompasses both precision and recall, so a good FMeasure score will be a balance of both of the basic retrieval metrics. However, metric also favors classifiers that yield high precision over classifiers that yield high recall. Therefore, classifiers with high precision with low recall can still yield high FMeasure scores.

FMeasure may also be defined in terms of precision and recall as follows:

$$f_\beta = \frac{(1 + \beta) * precision * recall}{\beta * precision + recall} \tag{3.4}$$

FMeasure is 0 when the number of articles retrieved is 0.

TREC11's T11SU is also used for comparing the performance of iScore with the work done in TREC11:

$$T11SU = \frac{2*\max(T11NU, -0.5) + 0.5}{1.5}$$
$$T11NU = \frac{2*|\text{Int Articles Retr}| - |\text{Unint Articles Retr}|}{2*|\text{Interesting Articles}|} \tag{3.5}$$

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

An ideal system would yield high FMeasure and T11SU scores overall and across time. In the experiments described in this dissertation, the classifiers are

evaluated several different ways since no one single test nor one single dataset can definitively identify the best classifier. The following tests are looked at:

1. The overall performance, averaged over all users in the collection, after the entire document collection is processed. The statistical significance of the difference between classifiers in this test is shown in Appendix D.

2. The overall average performance of classifiers, averaged over the bottom-k, the top-k, and all users in the collection, after the entire document collection is processed. The bottom-k and the top-k users are defined as the users for which iScore provided the best and worst recommendations, operating over Featureset A, as described in Appendix 4. For the Yahoo! News and Digg datasets, k is 10. For the tagger data set, k is 3. Featureset A is the original iScore featureset [PCB07a]. This test will show the performance of the classifiers for the very difficult and the very easy users to recommend for.

3. The cumulative average performance, averaged over all users in the collection, as documents are processed. This test will show any consistent overall performance differences among the classifiers, regardless of the collection size.

4. The average performance, averaged over all users in the collection, for the last 5,000 documents. This test will show the current performance of the classifiers.

Different configurations of iScore are evaluated in the experiments. The features are described in Table C.2 in Appendix C. iScore using naïve Bayes is denoted as NB, and iScore using naïve Bayes with feature selection is denoted as FSNB.

CHAPTER 4

The Basic iScore Architecture

4.1 iScore pipeline

News articles are processed in a streaming fashion, much like the document pro-
cessing 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 publi-
cation date, and the main content of the article. Articles are introduced to the
system in chronological order of their publication date. Once the system classifies
an article, an interestingness judgment is made available to the system by the
user.

The article classification pipeline consists of four phases, shown in Figure 4.1.
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 $I(d)$:

$$I(d) = C(f_1(d), f_2(d), ..., f_n(d)) \tag{4.1}$$

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
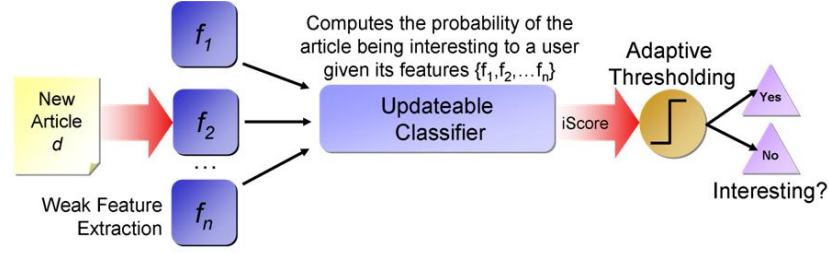
Figure 4.1: Article classification pipeline.

the next document in the pipeline.

4.2 Basic features for classification

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

4.2.1 Topic relevancy

Although an article that is relevant to a topic of interest may not necessarily be interesting, relevancy to such topics is a prerequisite for interestingness for a certain class of users. Five different methods is used to measure topic relevancy. These methods represent a selection of the best well-known information retrieval algorithms and are easily implementable.

The first method is the Rocchio adaptive learning method [Roc71]. Further discussion on the Rocchio algorithm is available in [Joa96], in which the author compares the Rocchio relevance feedback algorithm with its probabilistic variant and the standard naïve Bayes classifier.
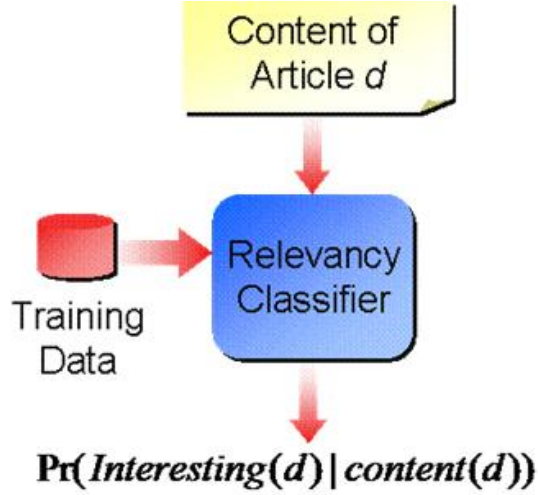
Figure 4.2: Topic relevancy.

A document is represented as a vector $\vec{d}$. 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 the experiments, tokens are stems produced by the Porter algorithm [Por80]. Stems occurring only once in the collection are discarded to reduce the feature space, which has been shown to improve classification time and results [YP97].

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

$$\vec{p} = \ \vec{p} + \vec{d} \ \ \text{if } d \text{ is interesting} \tag{4.2}$$

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:

$$cos(\vec{p}, \vec{d}) = \frac{\vec{p} \bullet \vec{d}}{|\vec{p}||\vec{d}|} \tag{4.3}$$

The second method for measuring topic relevancy is a variant of Rocchio by

[XYW02], which updates profiles as follows:

$$\vec{p} = \begin{cases} \vec{p} + \chi * \vec{d} & \text{if } d \text{ is interesting} \\ \vec{p} - \gamma * \vec{d} & \text{if } d \text{ is not interesting} \\ \vec{p} - \gamma' * \vec{d} & \text{otherwise and } \cos(\vec{p}, \vec{d}) < t \end{cases} \qquad (4.4)$$

The first two conditions are satisfied by user taggings. The third condition is for pseudo-negative documents, which have no taggings and its similarity with the profile is below a threshold. Good values (in TREC11) for χ, γ, γ', and t are 1, 1.8, 1.3, and 0.6, respectively [XYW02].

The other three methods for measuring topic relevancy use language models. An n-gram language modeling approach has been used for document classification [PSW03], which is a method that is used for finding another set of topic relevancy scores. Like naïve Bayesian classifiers, language-based modeling classifiers classify documents given the number of occurrences of grams (e.g., words or characters) in the document. Unlike naïve Bayes, which assumes that grams occur independently, language modeling classifiers assume that an occurring gram is dependent upon the last $n-1$ grams. In other words:

$$P(d) = P(g_1, g_2, ..., g_N) = \prod_{i=1}^{N} P(g_i | g_{i-n+1}, \ldots, g_{i-1}) \qquad (4.5)$$

where N is the number of grams in the document and g_i is the i-th gram in the document d. $P(g_i | g_1, \ldots, g_{i-1})$ can be estimated with Jelinek-Mercer smoothing [CG96].

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 the number of times the model is updated and compiled

is minimized while still being able to produce meaningful results. The models are compiled at regular intervals (i.e., every time there is an update to the model and on a daily basis). To avoid biasing the models from classifying articles as uninteresting (since there are an overwhelming number of uninteresting articles compared to interesting ones) and to reduce compilation time, the models are updated with all interesting articles, and updated with uninteresting articles if the number of uninteresting articles already used to update the model is less than the number of interesting article seen.

Using language models, three topic relevancy measurements are extracted for each document. The first measurement is $P(Int|d)$, using a 6-gram character model. Another measurement is $P(Int|d)$, using a uni-gram model where grams are tokens consisting of two words – equivalent to a naïve Bayesian classifier. The final measurement is the sample cross-entropy rate between the language model of interesting past articles and the current article, using a 6-gram character model. The Lingpipe documenation [Ali06] recommends a 6-gram model for models based on characters. This is commonly referred to as the binary language model classifier.

4.2.2 Uniqueness

Articles that yield little new information compared to articles already seen may not be interesting. In contrast, an article that first reports a news event may be interesting. Anomalous articles that describe a rare news event may also be interesting. For example, in [RJ05], interesting articles may be produced by rare collaborations among authors. Methods for outlier detection include using mixture models [Esk00], generating solving sets [ABP05] and using k-dimensional trees [CSM02], to identify outliers. Other more recent work by [PLT05] proposes

a new statistic based on a score process for determining the statistical significance of a putative signal that may be a small perturbation in a noisy experimental background. Work in network security, such as intrusion detection has already leveraged work in outlier detection. For example, [LV02] uses a k-nearest-neighbor search (kNN) for intrusion detection. The occurrence of system calls is used to characterize program behavior. A system call is treated as a word in a long document and the set of system calls generated by a process is treated as the "document."

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

$$f_{Cluster-Anomaly}(d) = 1.0 - \frac{\sum_{p \in P} \cos(\vec{d}, \vec{p}) size(p)}{\sum_{p \in P} size(p)} \tag{4.6}$$

After the article has been evaluated, the clusters are updated. If the similarity between an article and a cluster is above a threshold, then the article is added to the cluster. An article may belong to more than one cluster. If there are no clusters to which the document is similar to, then a new cluster is added to the list of clusters given the document's vector. 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 recently 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 docu-
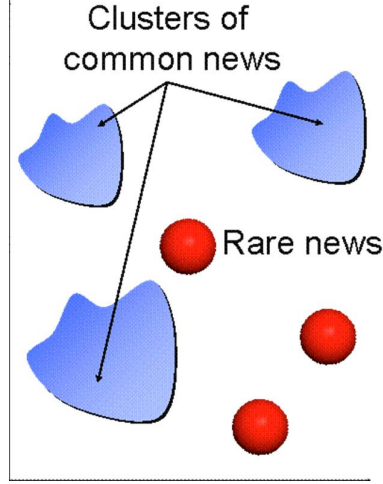
Figure 4.3: Uniqueness.

ments 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 outliers are accurately identified, and so the threshold is incremented by $growthRate$ (reaching a maximum threshold) whenever no new clusters have been added. In the experiments, the maximum threshold, $growthRate$, the initial threshold, and $maxCluster$ are set to 0.5, 0.01, 0.1, and 200, respectively.

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

$$f_{LM-Anomaly}(d) = \log(P(d|\text{documents seen before})) \qquad (4.7)$$

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

The second language model-based anomaly detection method measures the significance and the presence of new phrases. A background model is maintained

33

of all the documents previously seen and compare it with the language model of the current document. The sum of the significance of the degree to which phrase counts in the document model exceed their expected counts in the background model is computed. Only the top-10 phrases that exceed their expected counts are considered. A tri-gram model, where grams are word tokens, is used. Significance of each n-gram is based on the z-score [Ali06]:

$$z = \frac{numSuccesses - expectedSuccesses}{(numTrials * P(success) * P(1 - P(success)))^{1/2}} \quad (4.8)$$

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

Because language models are costly to compile, the models are compiled 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 few documents have been 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.

4.2.3 Source reputation

Source reputation estimates an article's interestingness given the source's past history in producing interesting articles. Articles from a source known to produce interesting articles tend to be more interesting than articles from less consistently interesting sources. Moreover, specific sources may specialize in particular topics

in which the user is interested. A news article's source may be its news agency or its author. In the experiments, the article's author(s) are used for the Yahoo! News and the TREC datasets, and the name of the host are used for the Digg and tagger datasets. the article's source reputation score is estimated as the average proportion of documents produced by the authors that were interesting in the past:

$$f_{Source-Rep}(d) = \frac{\sum_{a \in authors(d)} \frac{|\text{Int articles written by } a|}{|\text{Articles written by } a|}}{|authors(d)|} \qquad (4.9)$$

4.2.4 Writing style

Most work using the writing style of articles has mainly been for authorship attribution of news articles [LZC06] and blogs [KSA06]. Other than authorship attribution, changes in linguistic features over the course of a document have been used to segment documents as well [CA06]. Instead of author attribution and document segmentation, the same writing style features are used to infer interestingness. For example, the vocabulary richness [TB98] of an article should suit the user's understanding of the topic (e.g., a layman versus an expert). Also writing style features may help with author attribution, which can be used for classifying interestingness, where such information is unavailable.

A naïve Bayesian classifier is used and trained on a subset of the features from [CVX06], including syntactic, structural, lexical, word-based, and vocabulary richness features. Like the language models used in the topic relevancy measurements, the number of positive and negative articles used to update the classifier is balanced. The writing style score measured is:

$$f_{Writing-Style}(d) = P(Int|writingStyleFeatures(d)) \qquad (4.10)$$
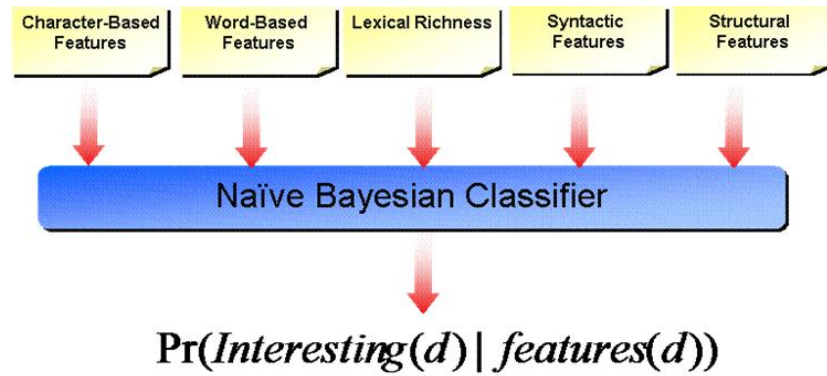
Figure 4.4: Writing style.

Feature
Total number of characters(C)
Total number of alphabetic characters/C
Total number of upper-case characters/C
Total number of digit characters/C
Total number of white-space characters/C
Total number of tab spaces/C
Frequency of letters

Table 4.1: Lexical features.

Feature	Description
Frequency of special characters	,@,#,$,%,¿&,*,-,‚,=,+,¿,¡,[,],/,\,|
Total number of words (M)	
Total number of short words/M	Words of less than four characters
Total number of characters/C	
Average word length	
Average sentence length	In terms of characters and words
Total different words/M	
Hapax legomena	Frequency of once-occurring words
Hapax dislegomena	Frequency of twice-occurring words
Yule's K measure	Yule's vocabulary richness
Simpson's D measure	Simpson's vocabulary richness
Sichel's S measure	Sichele's vocabulary richness
Burnet's W measure	Brune's vocabulary richness
Honore's R measure	Honore's vocabulary richness
Word length distribution/M	Count of words of differing lengths

Table 4.2: Word-based features.

37

Feature
Frequency of punctuations
Frequency of stop-words
Total number of lines
Total number of sentences
Total number of paragraphs
Number of sentences per paragraph
Number of words per paragraph
Includes quoted content

Table 4.3: Syntactic and structural features.

4.2.5 Freshness

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

$$f_{Freshness}(d) = \frac{1}{k} \sum_{d' \in \text{last } k \text{Int articles}} \log\left(Time(d) - Time(d') + 1\right) \qquad (4.11)$$

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

Figure 4.5: Freshness.

4.2.6 Subjectivity and polarity

The sentiment of an article may also contribute to an user's definition of interestingness. For example, "bad news" may be more interesting than "good news" (i.e., the polarity of the article). Or, subjective articles may be more interesting than objective articles. Polarity identification has been done with a dictionary [Mis05] and blog-specific features [Wie00]. Others have looked at subjectivity tagging, using various natural language processing (NLP) techniques [WWB04]. The density of subjectivity clues in the surrounding context of a word has been used to infer its subjectivity [Wie02] as well.

Four different features is maintained of this feature class: polarity, subjectivity, objective speech events, and subjective speech events. A speech event is a statement made by a person, such as a quotation. The Multi-Perspective Question Answering (MPQA) Opinion corpus [Wie02] is used to train 6-gram character language model classifiers. Each sentence in a document is classified to determine its polarity, subjectivity, and the presence of objective or subjective speech events, using these classifiers. The MPQA corpus is a news article collection from a variety of news sources annotated for opinions and other states, such as beliefs, emotions, sentiments, and speculations. For each document and each feature in this feature set, the following is measured:

$$f_{class}(d) = \frac{1}{|sentences(d)|} \sum_{s \in sentences(d)} P(class|s) \qquad (4.12)$$

39

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

4.3 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 [For04]. Consequently, an overall classifier must be incrementally updateable, robust against noisy and potentially useless features, and generate meaningful final scores for interestingness. Four classes of classifiers are evaluated: a naïve Bayesian classifier, non-incremental classifiers using a sliding window, temporal inductive transfer classifiers, and a linear combination using correlation for weights.

4.3.1 Naïve Bayesian classifier

A naïve 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, the following is found:

$$I(d) = P(Int|F(d)) \approx \frac{P(Int) \prod_f P(f(d)|Int)}{P(F(d))} \tag{4.13}$$

The probabilities can be estimated by maintaining statistics over feature values using kernel estimators [JL95].

4.3.2 Non-incremental classifiers

Three classifiers that are robust against irrelevant features, but are not incrementally updateable, are evaluated. 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, windowing classifiers are built 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 the experimental evaluations, 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, the C4.5 decision tree, built by the J48 algorithm [Qui93], is first evaluated. 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 prob-

ability of interestingness as a logistic function on a linear combination of features:

$$I(d) = P(Int|F(d)) = \frac{1}{1 + e^{-\sum_{f \in F} \lambda_f f(d)}} \tag{4.14}$$

A similar approach is taken in [CH92] to combine multiple ranking methods. A quasi-Newton method and ridge estimators are used to search for optimal values for λ_f [CH92].

In the experiments, logistic regression is more accurate than C4.5 under the windowing scheme, so logistic regression with bagging [Bre96] is evaluated as the third classifier. Bagging mitigates the instability of learning methods by building an ensemble of classifiers trained on randomly sampled instances from the training data. In the experiments, 100 ensemble classifiers are built.

4.3.3 Tix

A method used to address concept drift, called Temporal Inductive Transfer, or Tix [For06], is modified. 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 the experiments, logistic regression is used as the base induction algorithm, $P = 128$ classifiers, and $M = 1000$ articles.

4.3.4 Linear correlator

A linear correlator classifier that uses the correlation between a feature and interestingness is also studied. Intuitively, if a feature is highly correlated with interestingness, it should be weighted more in classifying the document. Unfortunately, this approach assumes that each feature is independent, ignoring the possibility that two features that perform poorly alone in predicting interestingness may perform well when combined together [GE03].

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

$$I(d) = \frac{\sum_f corr_f \sigma(f(d))}{\sum_f corr_f} \tag{4.15}$$

$$\sigma_f(d) = \frac{1}{1 + e^{-a_f(f(d)-t_f)}} \tag{4.16}$$

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. Feature values are assumed belong to two different normal distributions, one for interesting articles and one for uninteresting articles. The averages and standard deviations are incrementally maintained 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. The predicted true positive and true negative rates are computed, 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, a utility measure is computed, and the threshold with the greatest utility is selected. In the experiments, TREC's T11SU is used 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 the threshold's utility measure, which is proportional to the feature's predictive power, is used for the slope.

4.4   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, the threshold is dynamically adjusted 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, the efficacy of every threshold between 0 and 1 in increments of 0.01 is evaluated, without assuming that interesting and uninteresting articles are normally distributed. And in the case of ties between the utility measures, the threshold that provides the best separation between average iScores of interesting and uninteresting articles is selected. The utility measures

evaluated are T11SU and FMeasure f_β, where $\beta = 0.5$.

4.5   Initial Evaluation

In this section, the early evaluation of iScore is discussed. The experiments demonstrate the improvement of iScore over traditional information retrieval techniques. The goals of these experiments are to identify a good overall classifier and evaluate the overall effectiveness of the iScore framework.

4.5.1   Overall performance

In the following set of initial experiments, iScore is evaluated with the 35,265 articles dataset from Yahoo! News. The performance is evaluated of each overall classifier against several well known topic relevancy classifiers: Rocchio, the Rocchio variant, and the 6-gram character language modeling classifier. Each classifier is coupled with the adaptive thresholding mechanism, using FMeasure as the utility metric. Figure 4.6 shows the overall performance of the iScore classifiers compared to the baseline classifiers. The averages across RSS feeds, of precision, recall, and FMeasure are plotted in the graph. iScore with naïve Bayes outperforms the best baseline classifier (the language modeling classifier) by 20% in terms of FMeasure. 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 worse. However, the decision tree yields high recall at the cost of precision.

Naïve Bayes in iScore outperforms all the other classifiers used in iScore due to its ability to be incrementally updated quickly. The other classifiers can not
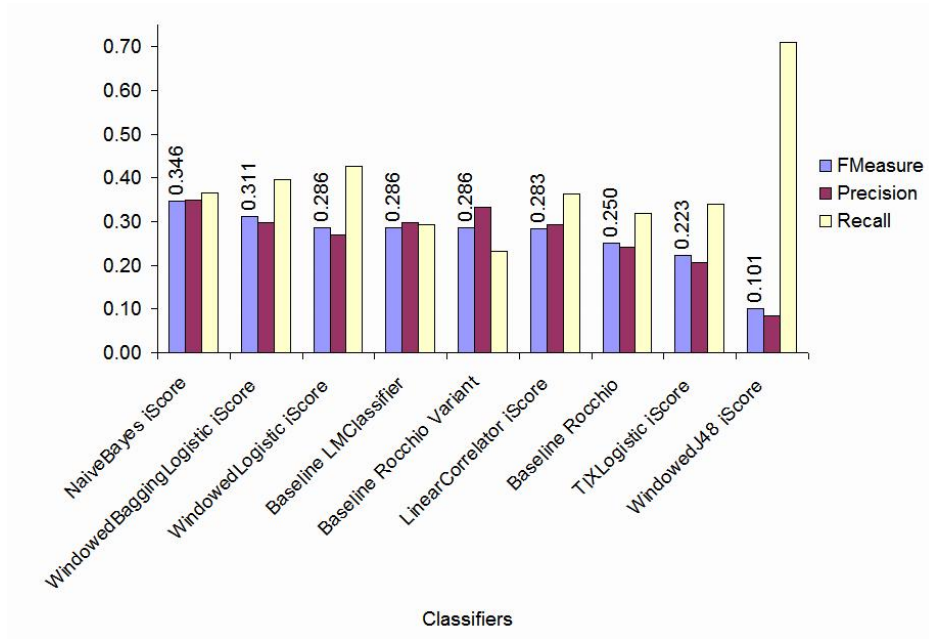
Figure 4.6: Overall performance of classifiers over the small Yahoo! News dataset. iScore with naïve Bayes outperforms the best baseline classifier by 20%.

be updated as every new document is processed due to computational and storage costs. The windowed classifiers have to operate over a sliding window of data items and are rebuilt at increasing intervals. Consequently, the windowed classifiers are trained with less data and are not necessarily up-to-date with the information about the last document processed.

Figure 4.7 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 worst 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
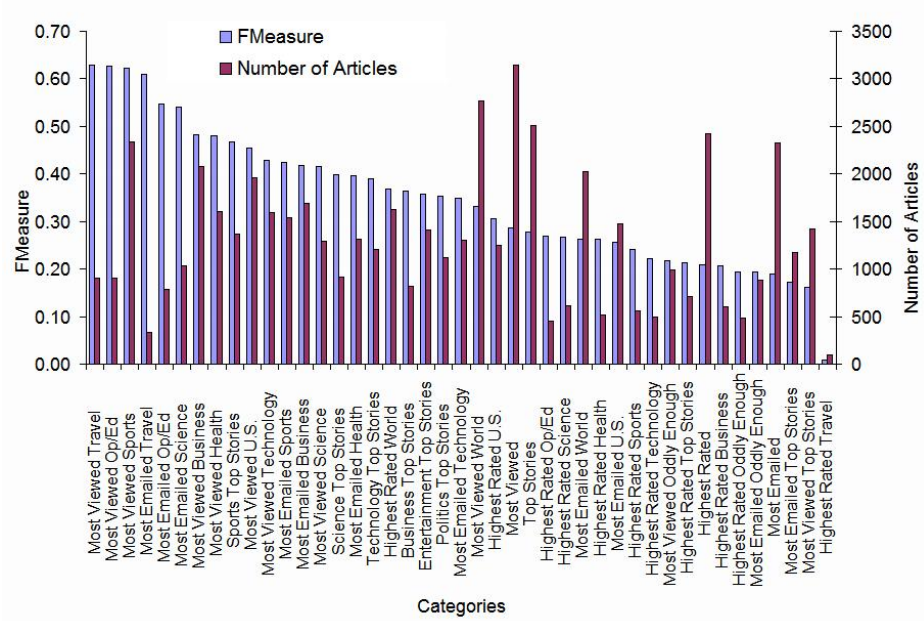
Figure 4.7: Performance of iScore (using naïve Bayes) in individual categories along with the number of articles in each category.

well for these feeds.

Since iScore uses some of the methods designed for the TREC task and topic relevancy is a prerequisite for interestingness, the iScore classifiers (coupled with the adaptive thresholder optimized for T11SU) are compared with the best filters from each participating group in TREC11 in Figure 4.8. 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! dataset. Logistic regression, logistic regression with bagging, Tix, and C4.5 yield the worst precision and FMeasure score but high recall.

The poor overall performance of iScore is due to the inclusion of features that are not useful for predicting relevancy for the TREC Adaptive Filter Task.
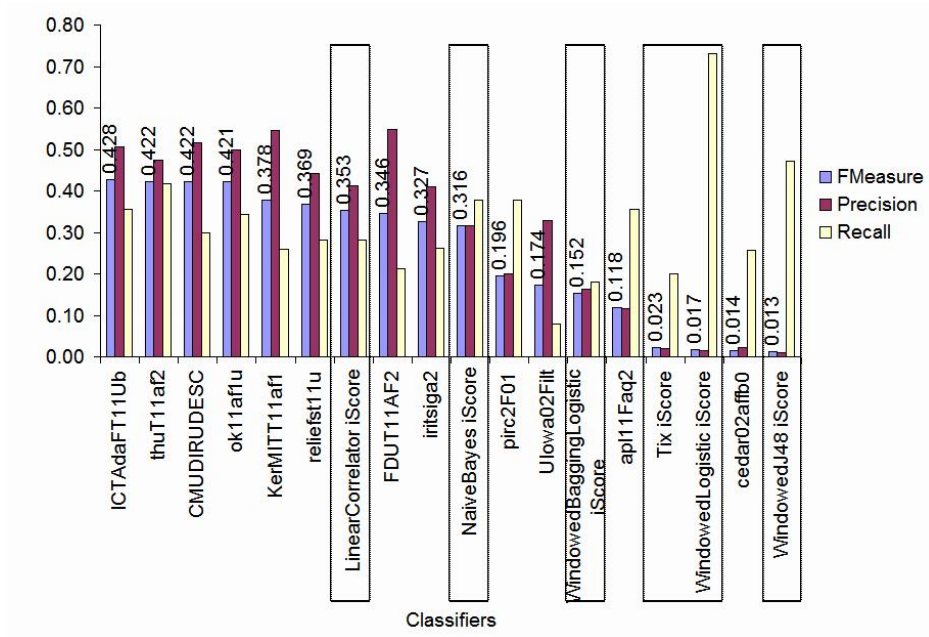
Figure 4.8: Overall performance of classifiers over the TREC articles. The iScore classifiers are outlined.

Figure 4.9 shows that the topic relevancy and source reputation scores are the only features correlated with relevancy in the TREC11 adaptive filter task. As expected, the Rocchio variant is the most correlated feature since it was the best performing filter in TREC11. The other iScore features included simply added noise to the classifier, so the naïve Bayes classifier required additional training to produce good recommendation results, which is shown in the next section. 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. If one were to compare TREC's correlation chart with those for the Yahoo!, tagger, and Digg datasets in Chapter 8, there is a stark contrast, where many more features are useful for those datasets. This

48
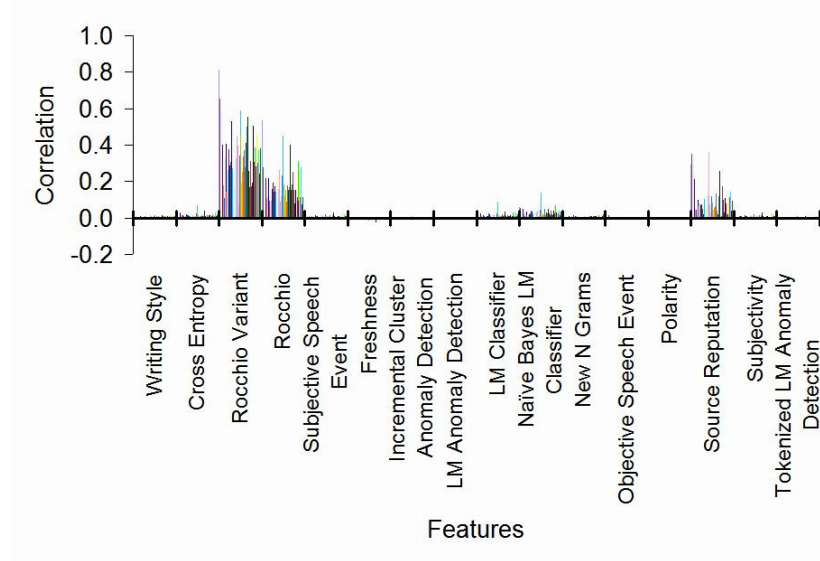
Figure 4.9: Feature correlation with relevancy in the TREC11 Adaptive Filter Task. Each color represents a different query.

further reinforces the proposition that the TREC Adaptive Filter Task does not fully address the "interestingness" recommendation problem.

4.5.2 Performance over time periods

Figure 4.10 shows the performance of each classifier over different time periods using the Yahoo! RSS articles. Each time period contains 5,000 articles. The best classifier used for iScore is the naïve Bayesian classifier, outperforming the best baseline classifier (the language modeling classifier) by 19.7% 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.
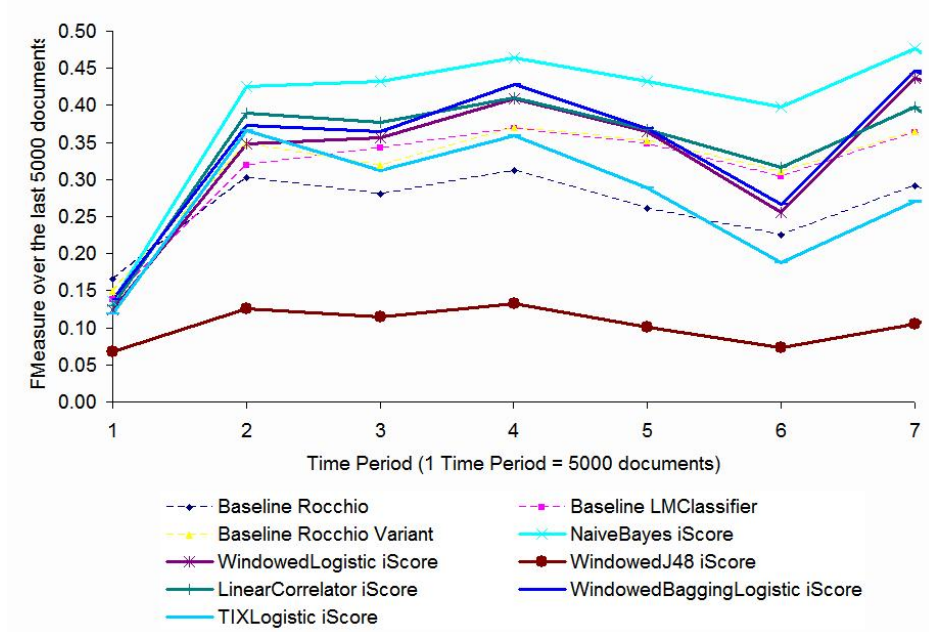
Figure 4.10: Performance of classifiers over time periods over the small Yahoo! News dataset. iScore with the naïve Bayes classifier outperforms the best baseline classifier by 19.7% on average.

The dip in performance in the sixth time period by most classifiers is due to concept drift (i.e., changes in the dataset over time) 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.

The T11SU performance of iScore is also compared with the best filters from TREC11 over time in Figure 4.11. Each time interval is a month's worth of articles. As in the overall performance analysis of iScore, logistic regression, logistic
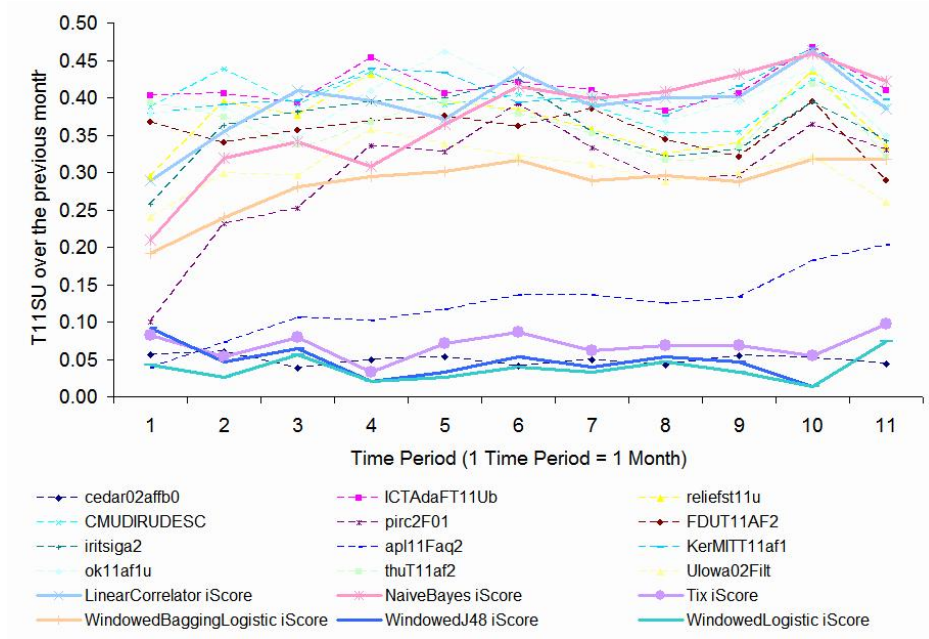
Figure 4.11: Performance of classifiers over time periods over the TREC articles.

regression with bagging, Tix, and the decision tree perform poorly; whereas, naïve Bayes and the linear correlator perform well. In the beginning periods, naïve Bayes performs poorly compared to the TREC filters and the linear correlator, but in the latter periods, it outperforms them. The lack of overall improvement in Figure 4.8 by iScore over the TREC filters and the slow increase in performance in Figure 4.11 are attributed to the additional learning layers in iScore. Also the multitude of useless features for the TREC task 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.

4.6 Discussion and summary

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, iScore is able to outperform most popular IR techniques in identifying interesting articles from Yahoo! RSS feeds by 20% overall and by 19.7% 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. Given the results of this evaluation, all further experiments involving iScore will use naïve Bayes as its overall classifier and the features discussed in this chapter.

CHAPTER 5

Multiple Topic Tracking

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 [Roc71]. A single profile vector $\vec{p}$ is maintained. For each document, the cosine similarity, or the cosine of the angle between the document vector $\vec{d}$ and the profile vector is measured.

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.

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 [SSS98]. 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 5.1 illustrates this problem. Although the cluster of all the interesting documents would contain interesting documents, it would also contain many uninteresting articles due to its size. If the user is interested in many orthogonal topics, then the encompassing cluster would be much larger and would also contain many more uninteresting articles as well.
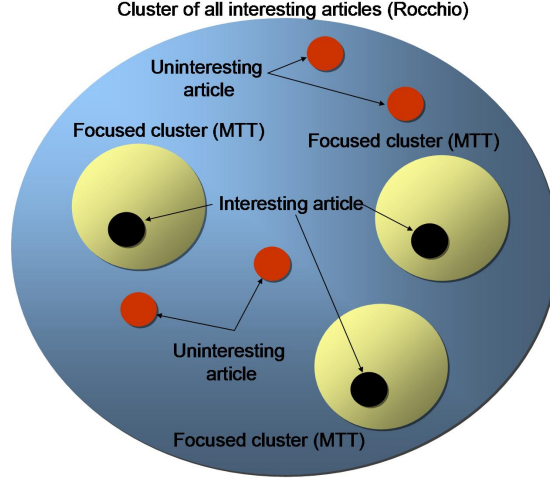
Figure 5.1: Failure of identifying relevant documents for multiple topics.

Instead, in Multiple Topic Tracking (MTT) [PCB07b], 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 5.1, 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 and topic detection and tracking (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.

5.1 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 [Por80] 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 $\vec{p_1}$ is created using the article's TF-IDF vector and added to P. Each subsequent article on the document stream with a document vector $\vec{d}$ is processed as follows:

1. Find the profile vector with the maximum similarity with $\vec{d}$. This profile represents the closest topic of interest to the document and is denoted as $\vec{p}_{max}$.

2. The score for a document is the thresholded similarity of the document vector $\vec{d}$ and $\vec{p}_{max}$. In other words:

$$f_{MTT}(d) = \begin{cases} 1 & \text{if } \cos(\vec{p}_{max}, d) > t_{\vec{p}_{max}} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The precision of $\vec{p}_{max}$ describes how well $\vec{p}_{max}$ can accurately identify interesting articles. The precision describes how interesting the user finds the topic that the profile vector represents. Another variant was looked at the product of the proportion of articles highly similar to $\vec{p}_{max}$ that were inter-

esting, but was found to be less useful than the scoring method described here.

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

$$\vec{p}_{max} = \vec{p}_{max} + \vec{d} \qquad (5.2)$$

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 $\vec{d}$ and $\vec{p}_{max}$ is less than the classification threshold $t_{classification}$, then $\vec{p}_{max}$ is updated as follows:

$$\vec{p}_{max} = \vec{p}_{max} - \gamma * \vec{d} \qquad (5.3)$$

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 [SMB97], 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. To reduce resource consumption and improve the quality of results, a method that discards profiles whose topics are no longer interesting is evaluated. Each profile vector can have an associated
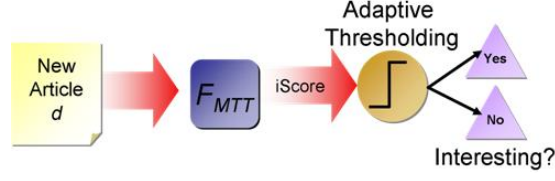
Figure 5.2: MTT evaluation pipeline.

precision for identifying interesting articles, which is defined as:

$$precision(\vec{p}) = \frac{|\text{Interesting articles with } \cos(\vec{p}, \vec{d}) > t_{classification}|}{|\text{Articles with } \cos(\vec{p}, \vec{d}) > t_{classification}|} \qquad (5.4)$$

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.

5.2 Parameter tuning

MTT is tuned with a collection of 35,256 news articles from all Yahoo! News RSS feeds, collected between June and August 2006. To find good values for the operating parameters: $t_{cluster}$, $t_{classification}$, $t_{precision}$, and γ, the quality of results produced by MTT with various parameters for the "Most Viewed Stories" RSS feed are evaluated. For simplicity and to evaluate MTT in isolation, the pipeline shown in Figure 5.2 is used instead of the complete iScore pipeline. The adaptive thresholder optimizes for FMeasure $\beta = 0.5$.

The effect of $t_{cluster}$ and γ on MTT is evaluated by varying $t_{cluster}$ while holding $t_{precision}$ and $t_{classification}$ at 0 and 0.6, respectively. The results are shown in Figure 5.3. As $t_{cluster}$ increases, articles are discouraged from clustering, resulting in much smaller and inaccurate clusters. Overall, FMeasure performance
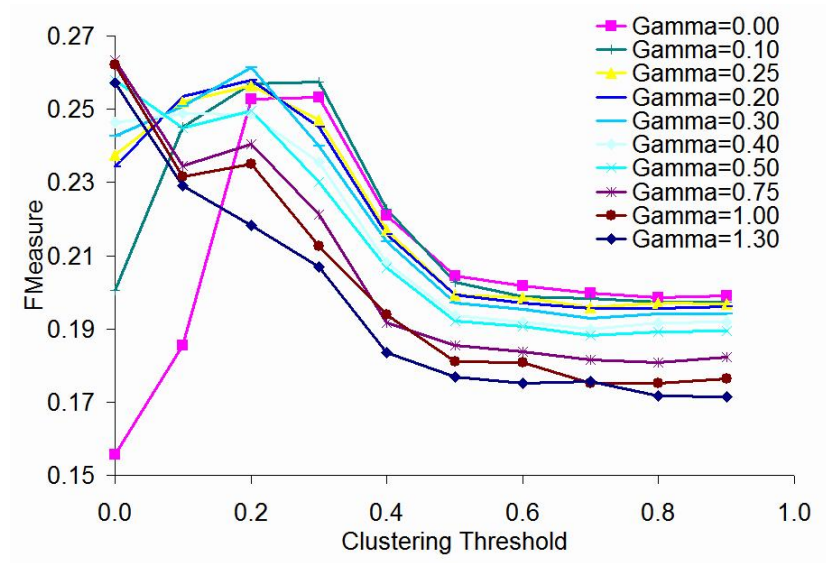
Figure 5.3: $t_{cluster}$ and γ for Most Viewed Stories from Yahoo! News.
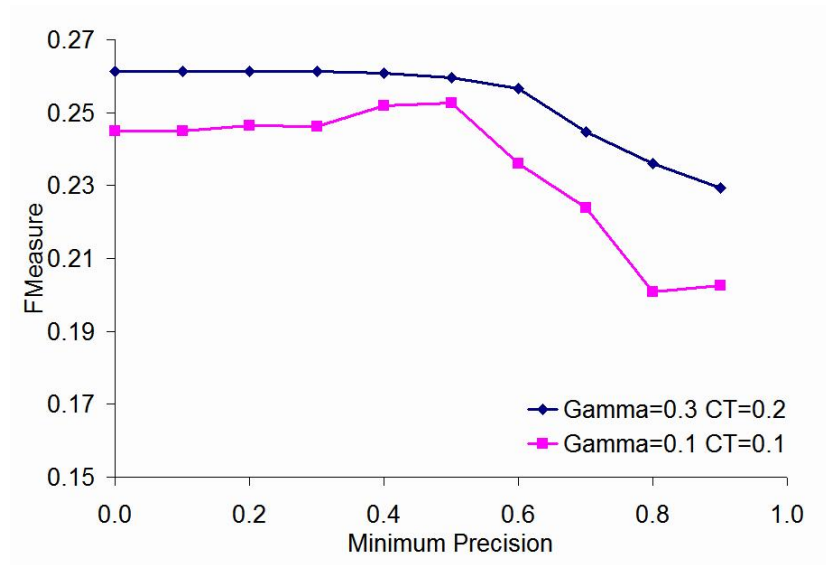
Figure 5.4: Minimum Precision for Most Viewed Stories from Yahoo! News.

increases initially as the clustering thresholding increases but drops drastically at much higher thresholding values greater than 0.3. The figure also shows that as γ increases, performance drops. However, an extremely low γ is not optimal. The figure shows that for the "Most Viewed Stories" feed, the best configurations for $t_{cluster}$ and γ are 0.1 and 0.1, respectively. Another good confgration for $t_{cluster}$ and γ are 0.2 and 0.3, respectively, as well.

Next, the effect of $t_{precision}$ on MTT is evaluated by varying $t_{precision}$ while holding $t_{classification}$ to 0.6 and $t_{cluster}$ and γ to the values found earlier. The results are shown in Figure 5.4. When the clustering threshold and γ are set to 0.3 and 0.2, respectively, performance decreases as the minimum precision increases. In this case, the best performance occurs when the minimum precision is 0 (i.e., no profiles are discarded). However, when the clustering threshold is 0.1 and γ is 0.1, performance peaks when the minimum precision is between 0.4 and 0.5.

5.3   Experimental results

In these set of experiments, MTT is evaluated using the parameter values ($\gamma = 0.3$, $t_{cluster} = 0.2$, $t_{classification} = 0.6$, and $t_{precision} = 0$) found previously with the "Most Viewed" feed from the small Yahoo! News collection. MTT is evaluated with the large Yahoo! News, tagger, and Digg, and datasets. In addition to the news recommendation datasets, MTT is evaluated on the TREC Adaptive Filter task.
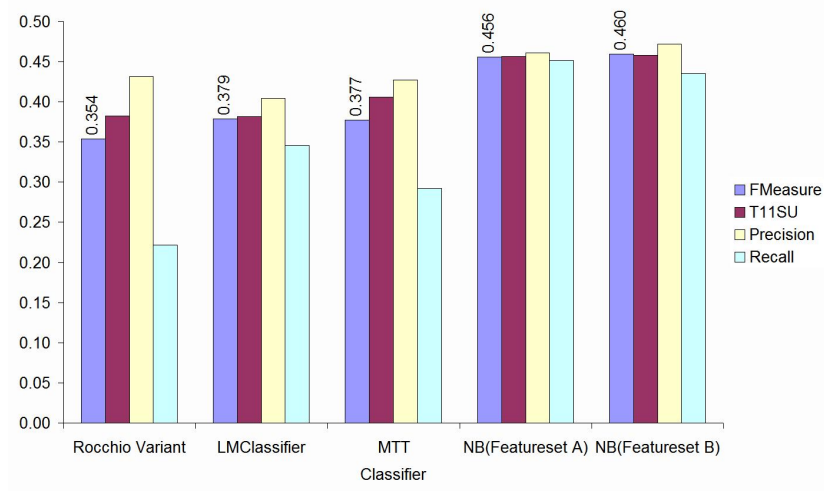
Figure 5.5: Average performance for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Yahoo! News dataset.

5.3.1 Yahoo! News

The first set of experiments evaluate MTT on the Yahoo! News dataset. Figure 5.5 shows the average FMeasure, T11SU, precision, and recall of various classifiers, including MTT and iScore with MTT. The figure indicates that MTT can outperform the Rocchio variant (which is the best performing filter in the TREC Adaptive Filter task) [XYW02] by 6.5% and performs as well as the language modeling classifier after processing the complete Yahoo! News collection. When MTT is added (Featureset B) to the original iScore featureset discussed in Chapter 4 (Featureset A), performance improves marginally.

Closer inspection of the results shows that there is performance improvement over the baseline classifiers and iScore by MTT. Figure 5.6 shows the average FMeasure of all, the bottom 10 performing, and the top 10 performing feeds/users. The figure indicates that MTT is 12.3% better than the language modeling classifier on the worst 10 performing feeds/users (when evaluated by iS-
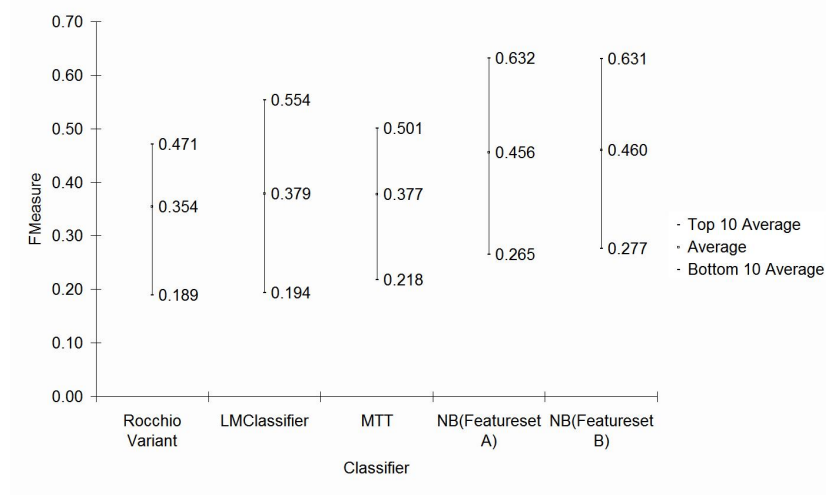
Figure 5.6: Bottom-10, top-10, and complete average FMeasure for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Yahoo! News dataset. MTT is 12.3% better than the language modeling classifier on the worst 10 performing feeds/users. When MTT is added to iScore, performance of the worst 10 performing feeds/users increases by 5%.

core). The performance of MTT is 9.5% less than that of the language modeling classifier on the best performing feeds/users. But when MTT is added to iScore, performance of the worst 10 performing feeds/users increases by 5% while the performance of the best performing feeds/users stays the same. This indicates that MTT can improve recommendation results when it is added to the iScore featureset.

Figure 5.7 shows the current FMeasure performance of the classifiers as documents are processed. The figure indicates that the advantage that MTT has is consistent regardless of the number of documents that have been processed. MTT has the advantage over the language modeling classifier in all time periods, except for time periods beyond 105, where MTT and the language modeling
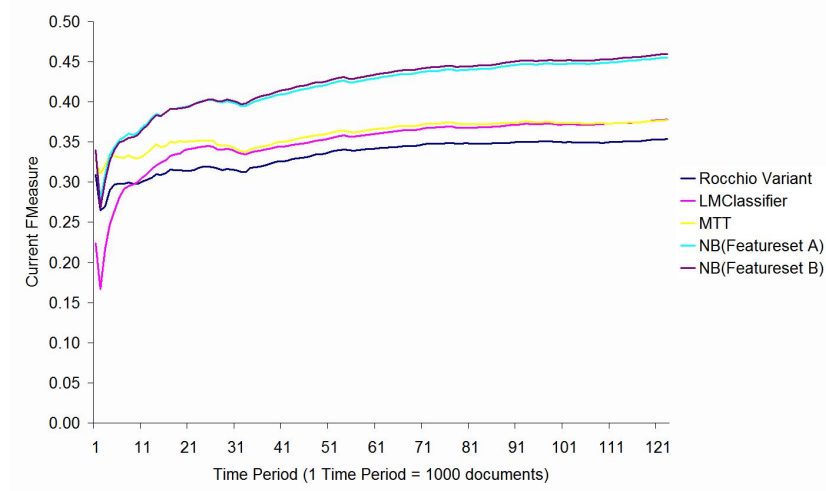
61

Figure 5.7: Cummulative FMeasure at specific periods for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Yahoo! News dataset.

classifiers are statistically tied. When MTT is added to iScore, the performance of iScore begins to improve after 21,000 documents have been processed. This indicates that it may take some time for MTT and the naïve Bayes classifier used as the overall classifier to learn enough to result in improved recommendation results.

Figure 5.8 shows the FMeasure performance of the classifiers over the most recent 5,000 documents processed. The intent of this evaluation is to show the current performance of various classifiers over the most recent documents. In this figure, it is not clear whether the language modeling classifier or MTT has the advantage in the first two-thirds of the experiment. In the latter third of the experiment, the language modeling classifier clearly performs better than MTT. It is also clear that after 25,000 documents have processed, iScore with MTT (Featureset B) has the advantage over iScore with only the original featuresets (Featureset A) in this evaluation test.
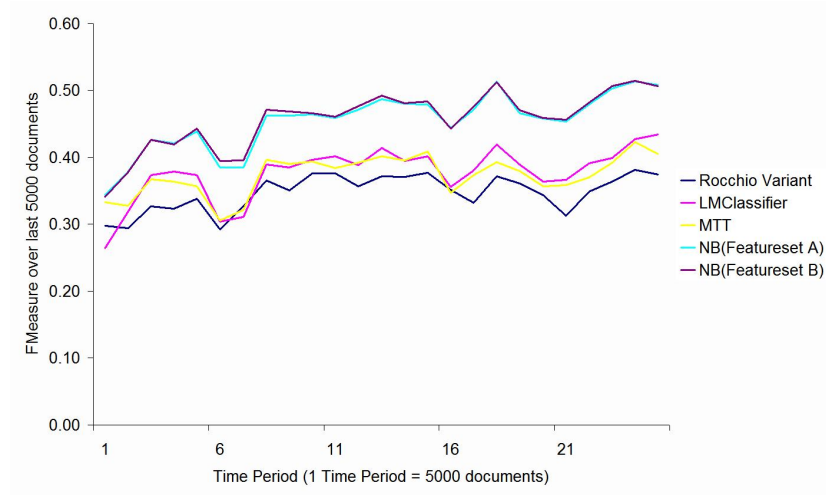
Figure 5.8: FMeasure for the 5,000 most recent documents for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Yahoo! News dataset. After processing 25,000 documents, iScore with MTT (Featureset B) has the advantage over iScore with only the original featuresets (Featureset A).
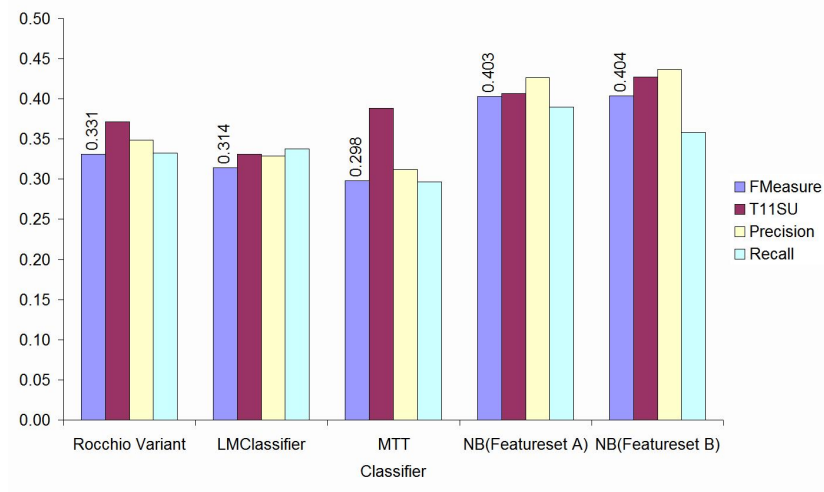
Figure 5.9: Average performance for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the tagger dataset. Performance marginally improves by 0.2% for FMeasure when MTT is added to iScore.

5.3.2 Tagger

The second set of experiments evaluate MTT on the tagger dataset. For the tagger dataset, MTT is worse than the Rocchio variant and the language modeling classifier in terms of FMeasure, as shown in Figure 5.9. On the other hand, in light of the T11SU metric, MTT performs better than both the Rocchio variant and the language modeling classifier. Additionally, when MTT is added to iScore, performance marginally improves by 0.2% for FMeasure. Performance also improves for T11SU when MTT is added to iScore.

Figure 5.10 also shows that MTT alone does not perform well for the tagger dataset. The chart shows that MTT worse than the langauge modeling classifier and the Rocchio variant for all three categories of users. However, the figure also shows that the performance of iScore improves when MTT is added to the featureset, for all three categories of users.
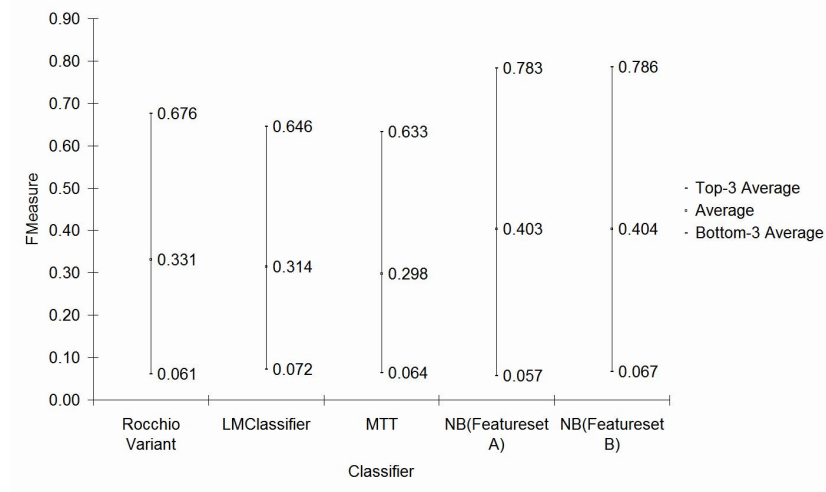
Figure 5.10: Bottom-3, top-3, and complete average FMeasure for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the tagger dataset.

5.3.3 Digg

The third dataset evaluates MTT on the Digg dataset. Figure 5.11 shows the average FMeasure, T11SU, precision, and recall of the Rocchio variant, the language modeling classifier, MTT, iScore with the original featureset, and iScore with MTT. The figure shows that although MTT does not perform as well as the Rocchio variant in terms of FMeasure, it has a significantly higher T11SU score due to its higher precision yet lower recall. When MTT (Featureset B) is added to iScore (Featureset A), the FMeasure performance of iScore improves by 3.1%. Although this new iScore configuration still has a lower FMeasure score as the Rocchio variant, it has a higher recall score.

Figure 5.12 shows the average FMeasure of all, the worst performing, and the top performing users. In the figure, MTT has approximately the same performance as the Rocchio variant for the worst performing users but significantly
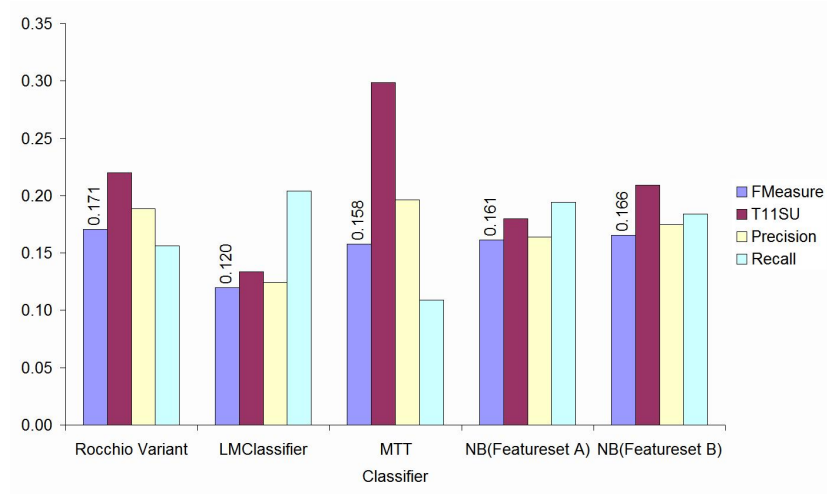
Figure 5.11: Average performance for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Digg dataset. When MTT (Featureset B) is added to iScore (Featureset A), the FMeasure performance of iScore improves by 3.1%. Although this new iScore configuration still has a lower FMeasure score as the Rocchio variant, it has a higher recall score.
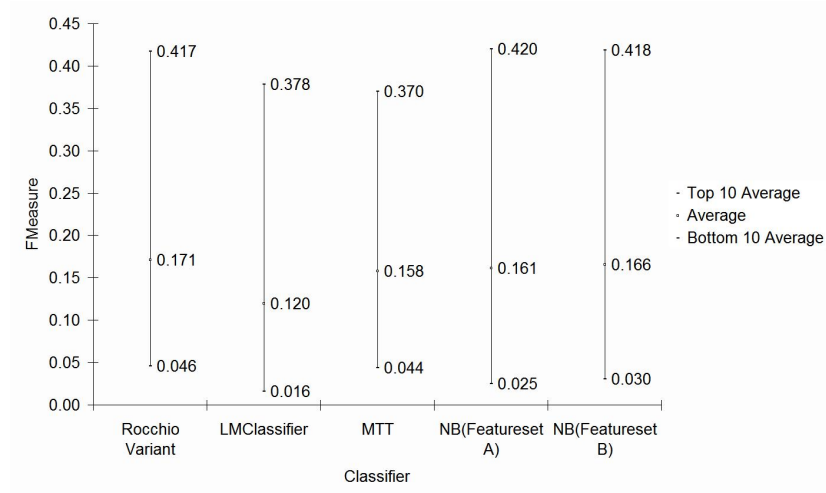
Figure 5.12: Bottom-10, top-10, and complete average FMeasure for the Rocchio Variant, LMClassifier, MTT, and iScorewith/without MTT on the Digg dataset.

poorer performance for the top performing users. When MTT is added to the iScore featureset, the FMeasure performance of the worst performing users improves marginally.

Figure 5.13 shows the FMeasure performance of the classifiers as documents are processed. At one point, MTT actually performs better than both iScore configurations (Featuresets A and B). But the figure indicates that the Rocchio variant has a better FMeasure score on average consistently.

However, in Figure 5.14, which shows the FMeasure performance of the classifiers for the 5,000 most recent documents processed, MTT has better performance for the 5,000 most recent document window than the Rocchio variant after 10,000 documents have been processed. The figure also indicates that the Rocchio variant gets its advantage in the overall FMeasure average in the early time periods, but in the later time periods, the iScore configurations, particularly iScore with MTT, gains the advantage.
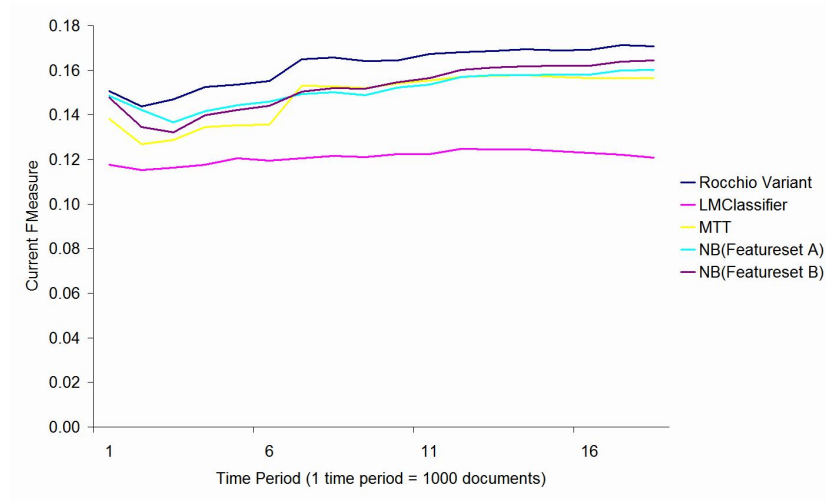
Figure 5.13: Current cumulative FMeasure at specific periods for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Digg dataset.
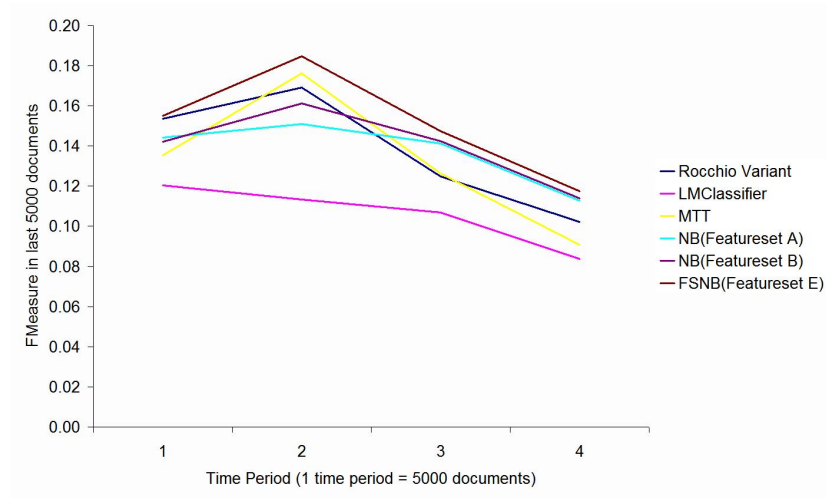
Figure 5.14: FMeasure for the 5,000 most recent documents for the Rocchio Variant, LMClassifier, MTT, and iScore with/without MTT on the Digg dataset. iScore has the advantage in the later time periods.

5.3.4 TREC Adaptive Filter

Although the TREC11 adaptive filter task is to retrieve all articles relevant to a query, regardless of its interestingness to a user, it is interesting 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 5.15. A majority of the TREC participants optimizedfor T11SU, so T11SU is used as the optimizing metric for the adaptive threshold. The same operating parameters found in the case study for the Most Viewed Stories feed are used. Figure 5.15 shows that MTT does not perform as well as the Rocchio variant (ICTAdaFT11ub) [XYW02]. However, when MTT is incorporated into iScore, FMeasure improves by 14%. According to [PCB07a], features other than topic relevancy features are not useful for identifying interesting articles to the TREC topics. The addition of irrelevant features causes the difference in performance between MTT alone and iScore with MTT while only improving iScore slightly when added as an additional feature.

Figure 5.16 shows the performance of iScore and MTT along with the top three adaptive filters from Figure 5.15. TREC only reports T11SU performance over time instead of FMeasure, so T11SU is shown in Figure 5.16. The figure shows that MTT performs much better over time than any of the other classifiers. Like Figure 5.15, Figure 5.16 does show improvement introduced by MTT into iScore over MTT alone and iScore without MTT for documents after time period 6.
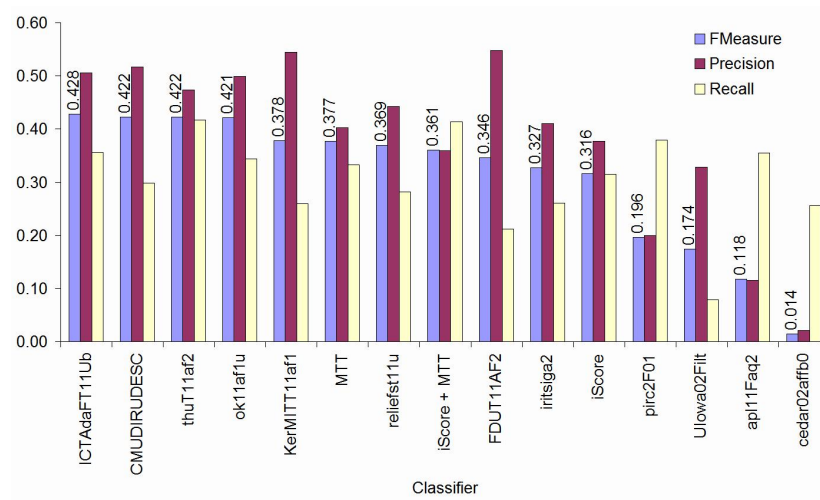
Figure 5.15: Average performance of the best filters, MTT, and iScore with/without MTT on the TREC Adaptive Filter task.
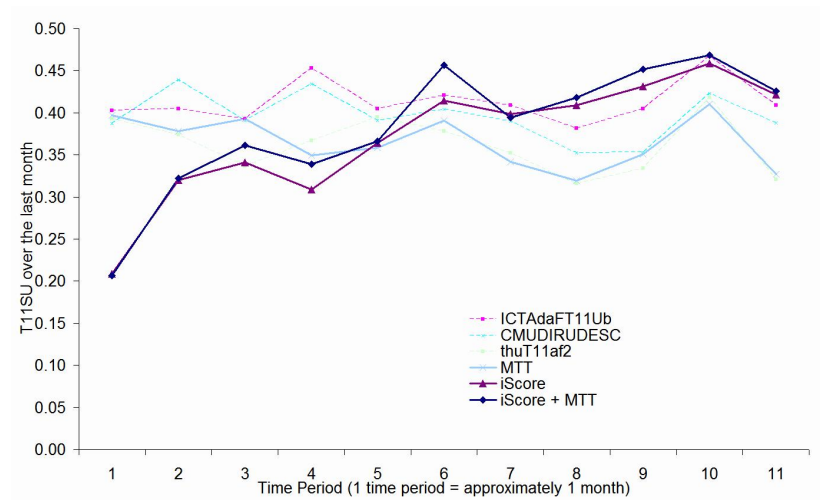
Figure 5.16: Performance for the last month's documents of the top performing filters, iScore, MTT, and iScore with MTT on the TREC Adaptive Filter task.

5.4   Discussion and summary

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 uninteresting 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, reasonably good operating parameters are found for MTT. Using these parameters, several evaluation experiments are performed. For the TREC adaptive filter task, iScore with MTT performs relatively well compared to the other filters from TREC11. MTT also outperforms the best filter over time as more documents are processed in terms of TREC11's T11SU metric. The inclusion of MTT in iScore improves its performance by 14%. For the news recommendation tasks, MTT has much higher T11SU scores than the baseline classifiers but with mixed FMeasure scores. With the exception of the tagger dataset, the inclusion of MTT improves iScore's overall FMeasure performance as well (0.8%, 0.2%, and 3.1% for the Yahoo! News, tagger, and Digg datasets, respectively).

However, the problem with MTT is that there are a multitude of parameters that must be tuned prior to running the algorithm. Also, these parameters may also be user-specific, so it would be impractical to tune for each user if the system is very large. This problem is addressed in the next chapter.

CHAPTER 6

Online Parameter Selection

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 [Roc71] and MTT [PCB07b], parameters often must be fine-tuned to a particular dataset through extensive experimentation. For example, in [XYW02], a Rocchio variant of the algorithm's performance depends extensively on the weight that is given to negatively labeled articles. In MTT, performance greatly depended upon γ, $t_{precision}$, and $t_{cluster}$. These parameters are determined through extensive trial and error experiments. If there are many different datasets that must be evaluated, this process is often tedious and expensive, leading many to simply fine-tune the parameters to one dataset and applying the parameters globally to all other datasets, which is almost certainly 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 behaviors. It is not feasible for a news recommendation engine to fine-tune parameters for every user because it

is very rare that validation data is available for fine-tuning until a user begins reading articles recommended by the system. Even if such a validation data was available, the task would be too time-consuming for it to be tractable if done on every user. To address this problem in news recommendation, the following contributions are made [PCB08b]:

1. Users are shown to have different learning/reading behaviors when evaluating the interestingness of news articles.

2. Instead of using static parameters, several different parameter configurations are evaluated simultaneously in a simple IR algorithm so that similar or better recommendation results can be achieved compared to more complex information retrieval algorithms (e.g., language modeling classifiers) and algorithms that require fine-tunining (e.g., MTT and Rocchio).

3. By tailoring parameters specifically to an user instead of using an "one-size-fits-all" set of parameters, better recommendation results can be achieved.

6.1 Rocchio

The Rocchio algorithm, first introduced in [Roc71], 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 $\vec{p}$ is updated as the following:

$$\vec{p}_{new} = \alpha * \vec{p}_{orig} + \chi * \sum_{d \in Rel} \vec{d} - \gamma * \sum_{d \notin Rel} \vec{d} \qquad (6.1)$$

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 TFIDF vector of an article. The set REL is the set of all relevant or positively tagged articles. The vector $\vec{p}_{new}$ is the TFIDF vector of the query profile. The vector $\vec{p}_{orig}$ is the TFIDF vector of some search query string. In a text filtering setting, there is often no initial user-query so $\alpha * \vec{p}_{orig}$ is ignored, simplifying the Rocchio formulation to the weighted sum of relevant documents and irrelevant documents. The Rocchio formulation can be incrementally computed as the following:

$$\vec{p}_{new} = \begin{cases} \vec{p}_{old} + \chi * \vec{d} & \text{if } d \in Rel \\ \vec{p}_{old} - \gamma * \vec{d} & \text{if } d \notin Rel \end{cases} \tag{6.2}$$

All negative components of the resulting profile are assigned a zero weight. 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 $\vec{p}$ as defined in Equation 4.3.

Other variations on Rocchio include the use of query zoning [SSS98] 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, [XYW02] uses different weights for negatively labeled documents and unlabeled documents. In [PCB07b], 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.

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 presetting 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 [CZ02], the lower bound of the number of mistakes that Rocchio will make in different scenarios was studied. In [Joa96], the connection between Rocchio and probabilistic classifiers, such as nave Bayes, was identified.

6.2   eRocchio

Given the shortcomings of existing information retrieval (IR) algorithms, such MTT, Rocchio, and its variants, that require fine-tuning parameters before the algorithms are run on live data, a different approach is taken. Rather than predetermining the weighting scheme in the Rocchio formulation in Equation 6.2, multiple instances of the Rocchio formulation are evaluated in parallel, each with a different weighting scheme. In Equation 6.2, there are two unknown parameters χ and γ, the relative weights for positively labeled articles and for negatively labeled articles, respectively. However, because γ is a weight relative to χ, multiple γ-values are evaluated simultaneously while holding χ to 1. This scheme is called 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 6.1. In the experimental evaluation, all possible γ-values between 0 and 2, inclusive, in intervals of 0.01, are evaluated. Because the cosine similarity between the query profile and the document is a real number bounded between 0
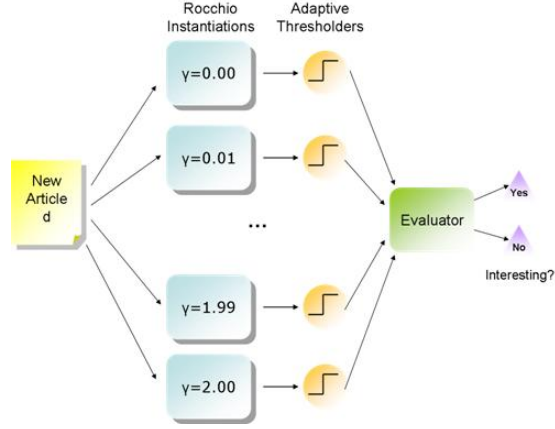
Figure 6.1: eRocchio classification pipeline.

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 thresholder. After each adaptive thresholder has generated a binary score from its corresponding Rocchio instantiation's generated similarity score, the evaluator must generate a final answer. The best Rocchio instantiation and its corresponding threshold are chosen by selecting the Rocchio instantiation and the threshold combination that has had the best utility measure up to that point. In evaluations, FMeasure F_{β} is used, where β=0.5, weighting precision twice as much as recall, which is consistent with the utility measure

76

used in the TREC Adaptive Filter Task [RS02].

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.

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 6.2.

2. The FMeasure statistic for each instantiation is updated.

3. The adaptive threshold for each instantiation is updated by updating the FMeasure 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
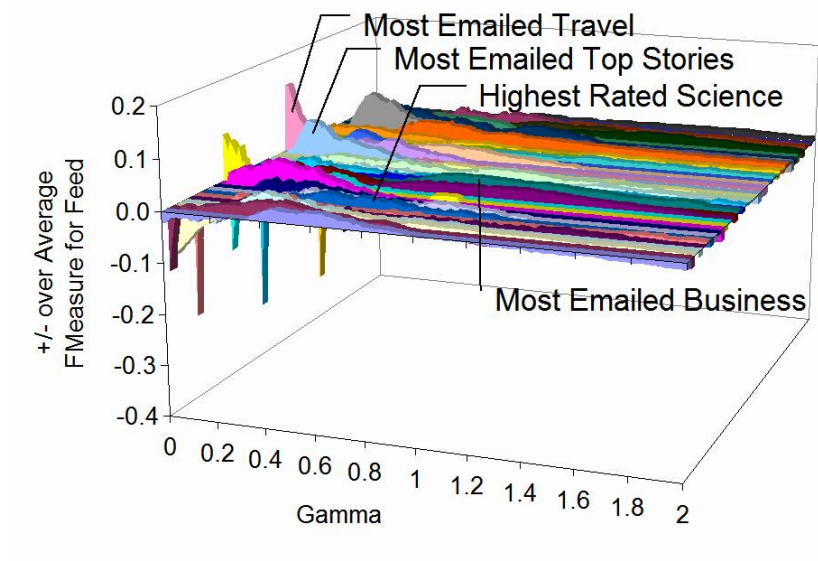
Figure 6.2: Each area curve is the normalized final FMeasure of each instantiation. A curve for each feeds from the Yahoo! News collection are shown.

large, with the availability of large-scale cluster computing, the multiple instantiations may be evaluated in parallel.

6.3 User variations

The variations of parameters among users with three datasets are studied. The first dataset is a set of 123,653 news articles from the Yahoo! News RSS feeds. The second dataset consists of articles collected from volunteer news readers that tag articles as they read their daily news on the web. The third dataset is the Digg collection.

Figures 6.2, 6.3(a), and 6.3(b) 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 FMeasure of each instantiation. The final FMeasure statistic
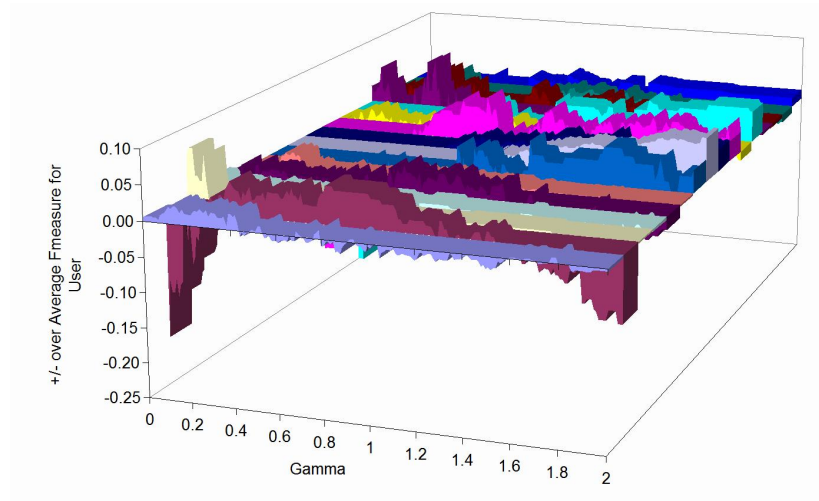
(a) Tagger dataset.
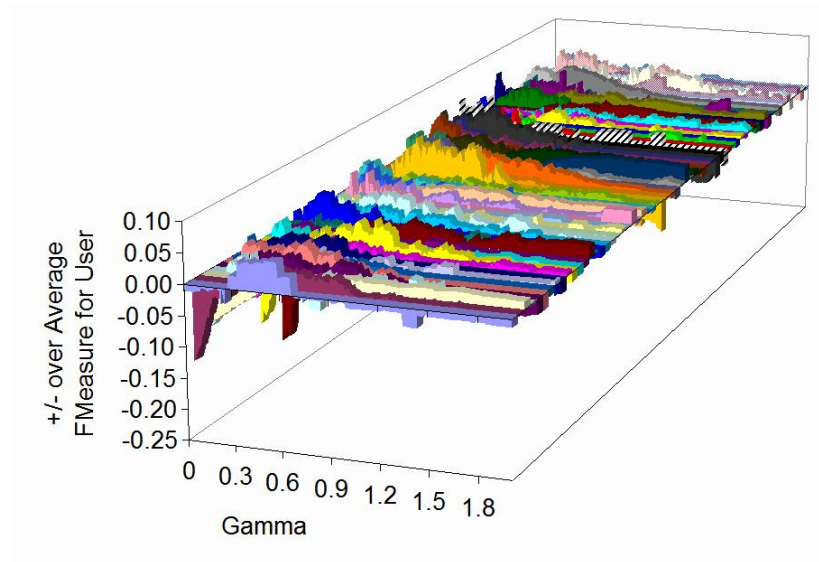
(b) Digg dataset.

Figure 6.3: Each area curve is the normalized final FMeasure of each instantia-tion. A curve for each user from the tagger and Digg collections are shown.
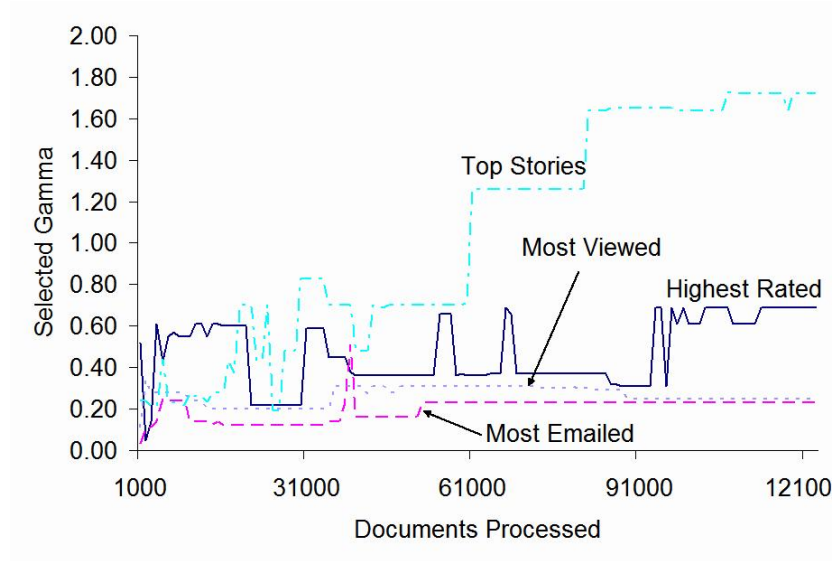
Figure 6.4: Selected γ-value over time for a select number of feeds from the Yahoo! News collection.

has been normalized so that the graph shows the deviation from the average final FMeasure for a given feed/user. Figure 6.2 shows the normalized FMeasure 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" 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 6.4 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

80

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 6.3(a) shows the normalized FMeasure 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. Figure 6.3(b) shows even more variation of the optimal γ weight in the Digg dataset.

6.4 Experimental results

In these sets of experiments, eRocchio's recommendation performance is evaluated with the large Yahoo! News, tagger, and Digg datasets. In addition to the news recommendation datasets, eRocchio is evaluated on the TREC Adaptive Filter task.

6.4.1 Spacing

The effect of the spacing between the evaluated γ-values on recommendation performance is first looked at. This experiment is run on the large Yahoo! News dataset. The FMeasure of eRocchio for various spacings between γ-values is shown in Figure 6.5. The figure shows that performance peaks when the spacing is 0.01. The performance drops as the spacing increases. This is the expected behavior because having smaller spacing between the evaluated γ-values allows eRocchio to evaluate more data points in identifying the optimal learning behav-
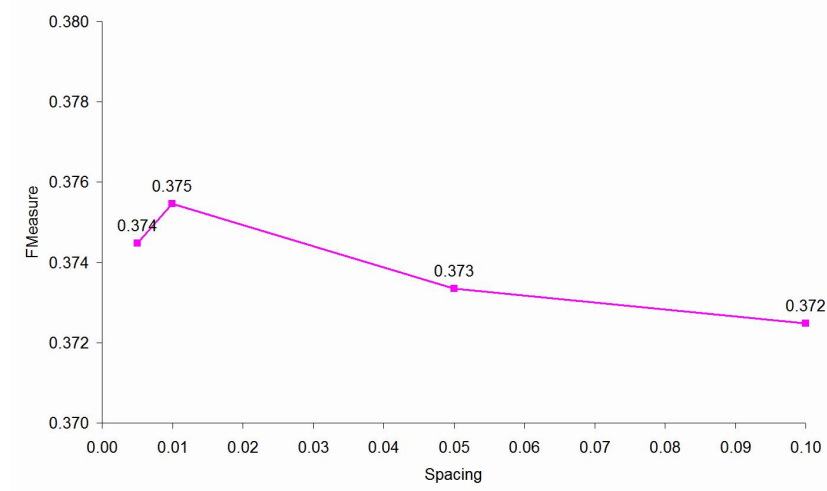
Figure 6.5: Average FMeasure of eRocchio on the Yahoo! News dataset for various spacings between γ-values.

ior.

6.4.2 Yahoo! News

The first set of experiments evaluate eRocchio on the Yahoo! News dataset. The spacing between γ-values evaluated in this experiment is 0.01. Figure 6.6(a) shows the average FMeasure, T11SU, precision, and recall of the Rocchio variant [XYW02] (which was the best performing filter in the last run of the TREC Adaptive Filter task), a language modeling classifier, MTT, and eRocchio. The figure shows that eRocchio has very similar performance to MTT and the language modeling classifier although it is a much simpler algorithm and requires no parameter tuning. In Figure 6.6(b), when eRocchio is added to iScore (Featureset C), it performs slightly better than the original featureset (Featureset A) and performs slightly worse than iScore with MTT (Featureset B). iScore with both MTT and eRocchio (Featureset D) performs 0.4% better than iScore with MTT

in terms of FMeasure, and has the best precision over all the classifiers with some loss in recall.

Figure 6.7(a) shows the average FMeasure of performance of all, the top 10, and the bottom performing feeds for several classifiers on the Yahoo! News collection, such as the Rocchio variant, a language modeling classifier, MTT [PCB07b], and eRocchio. 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 FMeasure 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 6.7(b) shows several different iScore configurations evaluating the Yahoo! News collection. The basic iScore configuration (iScore) includes all the features detailed in Chapter 4, including Rocchio, the Rocchio variant, and the language modeling classifier. The figure shows that when MTT (Featureset B) or eRocchio (Featureset C) is added to the original iScore features (Featureset A), 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 (Featureset C), performance remains relatively the same. And when both eRocchio and MTT are part of the featuresets (Featureset D), performance also stays relatively the same with a slight improvement on the worst 10 feeds.

Figure 6.8(a) shows the FMeasure performance over time as documents are
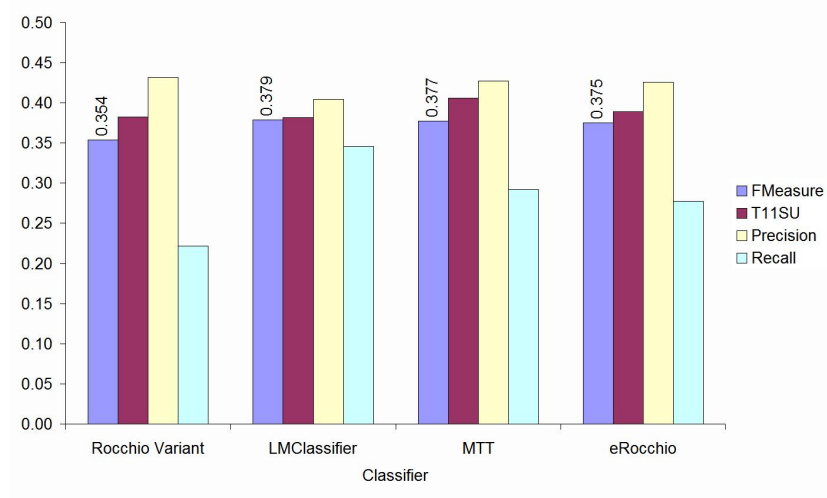
(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
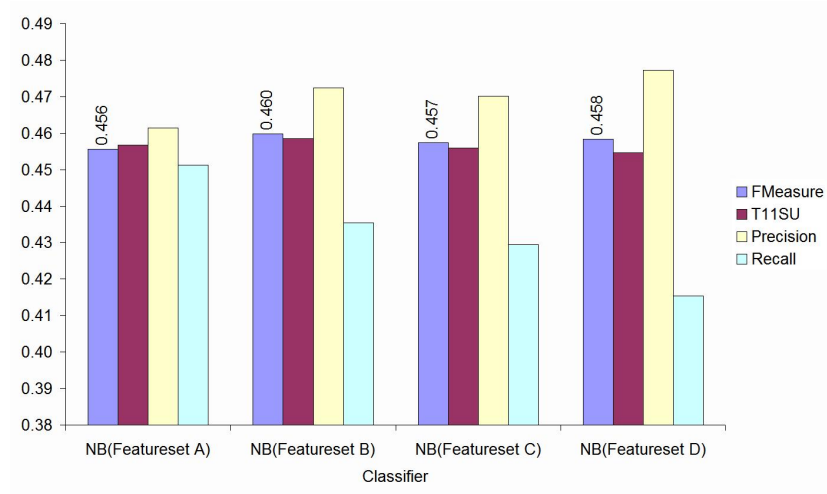
(b) iScore with different featuresets. iScore with both MTT and eRocchio performs 0.4% better than iScore with MTT in terms of FMeasure, and has the best precision over all the classifiers with some loss in recall.

Figure 6.6: Overall performance of the Rocchio variant, LMClassifier, MTT, eRocchio, and iScore with different featuresets on the Yahoo! News dataset.
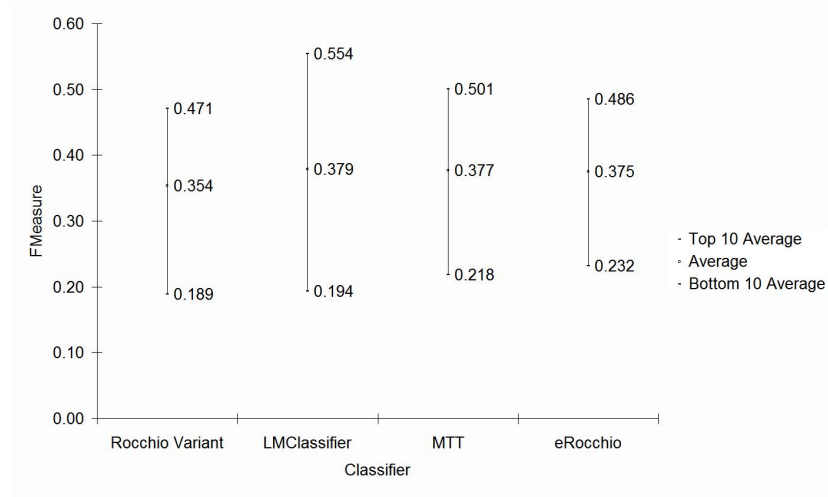
(a) Rocchio variant, LMClassifier, MTT, and eRocchio. eRocchio out-performs the Rocchio variant by a significant 2.1 FMeasure points (6% improvement). Also eRocchio performs better than all the other classifiers for the bottom 10 feeds, which are the most difficult to recommend articles for.
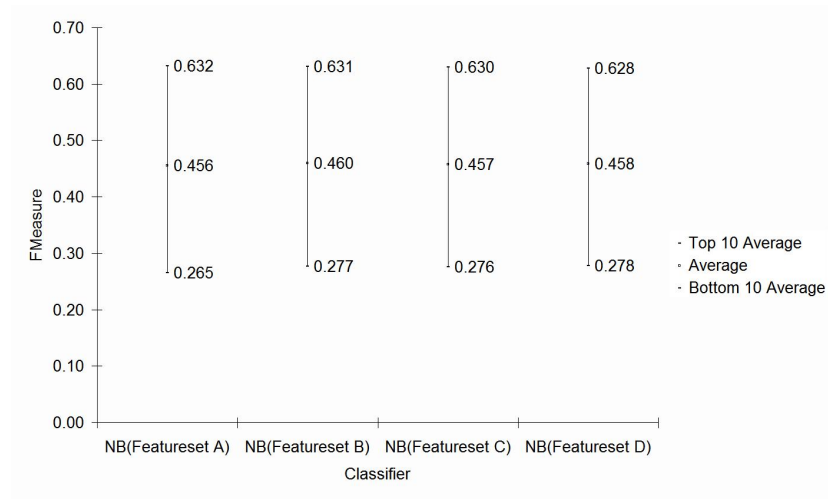
(b) iScore with different featuresets. When MTT is replaced with the much simpler eRocchio (Featureset C), performance remains relatively the same.

Figure 6.7: Bottom-10, top-10, and complete average FMeasure for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the Yahoo! News dataset.

(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
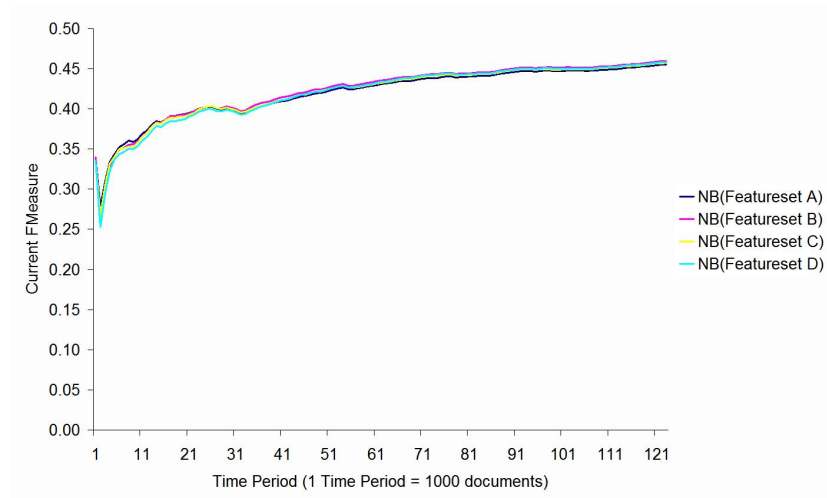
(b) iScore with different featuresets.

Figure 6.8: Current cumulative FMeasure performance at specific periods for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the Yahoo! News dataset.

processed. Although MTT outperforms eRocchio consistently, eRocchio outperforms the language modeling classifier in most time periods and has comparable performance with MTT despite its simiplicity and lack of parameter tuning. It is not clear in Figure 6.8(b) which iScore configuration performs the best. Although, the figure indicates that iScore with MTT and eRocchio and iScore with MTT are generally better than the other configurations.

Figure 6.9(a) shows the average FMeasure performance of the same classifiers over the last 5,000 documents. The figure shows that eRocchio performs as well as most of the baseline classifiers, with the best performance in the middle periods. In the later periods, however, the language modeling classifier has the advantage. Like the cumulative counterpart of this figure, it is not clear which iScore configuration is the best performance in Figure 6.9(b).

6.4.3   Tagger

The second round of experiments evaluate eRocchio on the tagger dataset. The spacing between γ-values evaluated in this experiment is 0.01. Like in the Yahoo! News dataset, eRocchio has better performance than the best baseline classifier in the tagger dataset. Where MTT dominates as the best baseline classifier in the Yahoo! News dataset, the Rocchio variant dominates in the tagger dataset. And eRocchio performs the Rocchio variant by 0.8%. Figure 6.10(a) shows that eRocchio can outperform the Rocchio variant, the language modeling classifier and MTT due to its higher precision and recall. Figure 6.10(b) shows that when eRocchio is added (Featureset C) to iScore, performance improves by 1.7%. However, when both MTT and eRocchio are added to iScore (Featureset D), performance drops, due to lower precision and recall, but it's T11SU score is comparable to that of when only eRocchio is added to iScore (Featureset C).

(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
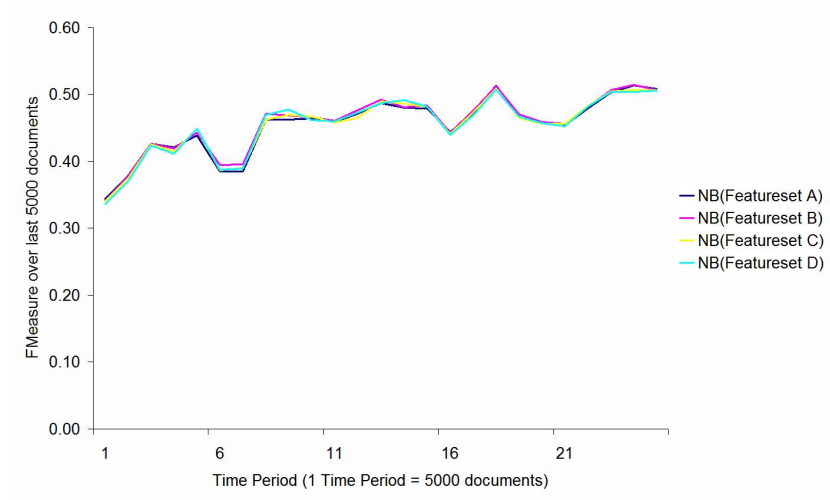
(b) iScore with different featuresets.

Figure 6.9: FMeasure for the 5,000 most recent documents for the Rocchio variant, LMClassifier, MTT, eRocchio, and iScore on the Yahoo! News dataset.
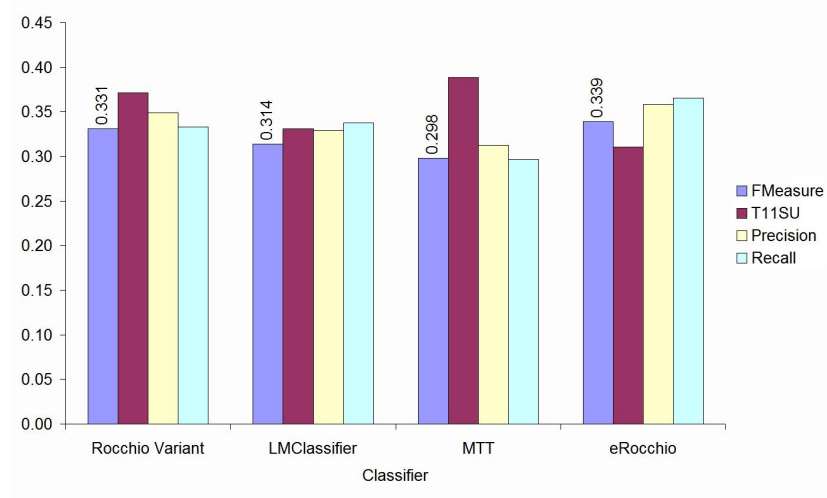
(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
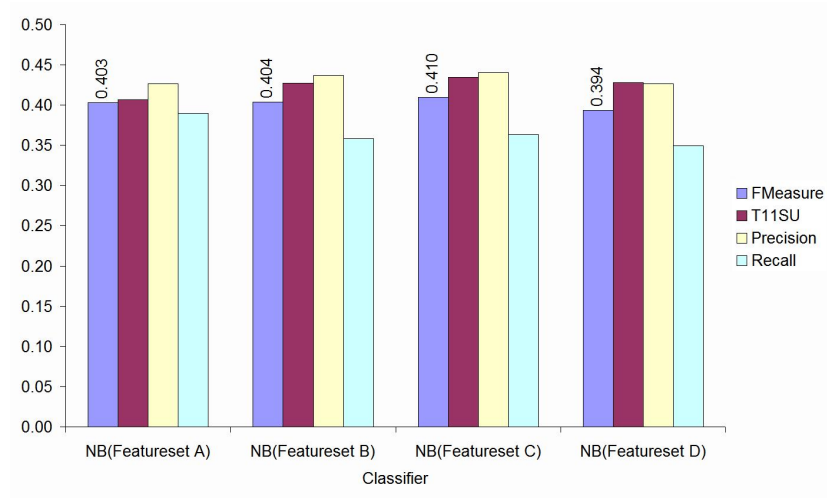
(b) iScore with different featuresets.

Figure 6.10: Overall performance of the Rocchio variant, LMClassifier, MTT, eRocchio, and iScore with different featuresets on the tagger dataset. eRocchio performs the Rocchio variant by 0.8%.

Figure 6.11(a) shows the average FMeasure performance of the same classifiers on the tagger dataset. The figure also shows the worst 3 and best 3 performing users. In this dataset, in contrast to the Yahoo! News dataset, the language modeling classifier and MTT do not perform as well as the Rocchio variant. The figure shows that while eRocchio performs better for the top 3 and the bottom 3 users performing users. On average, eRocchio performs as well as the Rocchio variant (using parameters recommended by [XYW02]), despite eRocchio's lack of parameter tuning that is required of the Rocchio variant.

Figure 6.11(b) also shows the same iScore configurations for the volunteer tagger dataset. The figure shows that the inclusion of eRocchio into iScore (Featureset C), recommendations for the hardest users improve while dropping only slightly for the easiest users. When both MTT and eRocchio are added to iscore (Featureset D), performance for the hardest users is the highest for the classifiers examined. The figure also shows that by adding MTT to the featureset that already includes eRocchio (Featureset D), performance improves slightly for the easiest users as well, compared to the performance of having eRocchio and the original iScore features alone.

6.4.4 Digg

The third datset evaluates eRocchio on the the Digg dataset. The spacing between γ-values evaluated in this experiment is 0.01. Figure 6.12(a) shows the average FMeasure, T11SU, precision, and recall of eRocchio compared to several other classifiers. The figure shows that eRocchio attains an 8% higher FMeasure score than the Rocchio variant, one of the baseline classifiers due to its higher precision and recall. This indicates that online parameter selection is significantly better than the static parameter selection for the Digg dataset. In Figure
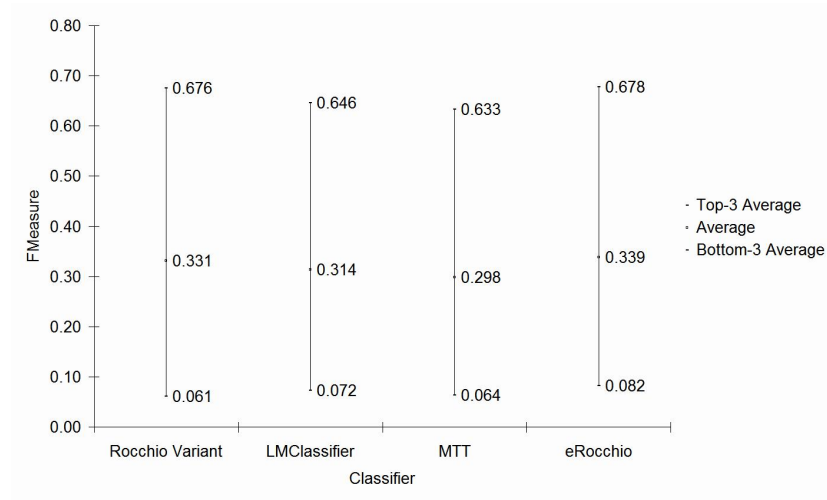
(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
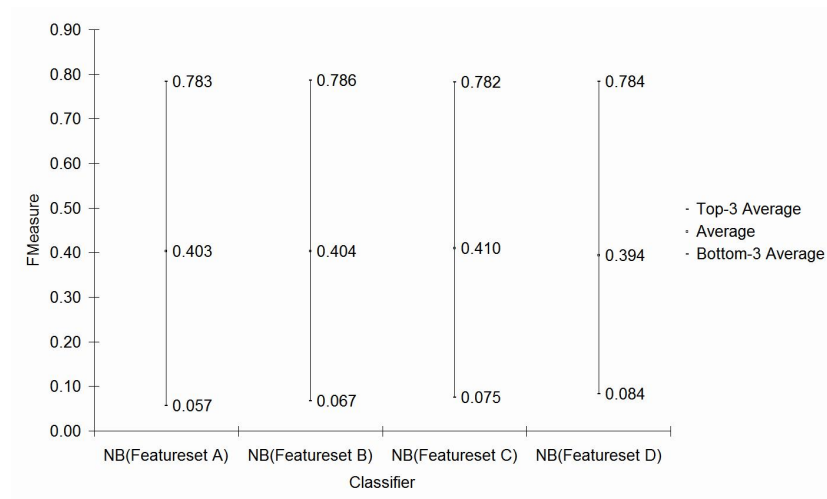
(b) iScore with different featuresets.

Figure 6.11: Bottom-3, top-3, and complete average FMeasure for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the tagger dataset.

6.12(b), when eRocchio (Featureset B) is included in the iScore featureset, iScore's FMeasure performance improves by 5.6%, but is still lower than that of eRocchio alone.

Figure 6.13(a) shows the average FMeasure of all, the bottom 10, and the top 10 performing users. The figure indicates that eRocchio performs better for recommendations to both the easiest and most difficult users. In Figure 6.13(b), when eRocchio is added to the iScore featureset (Featureset C), average FMeasure performance of iScore improves, making the most significant gains with the easiest users to recommend for. The addition of eRocchio (Featureset C) even outperforms iScore with MTT (Featureset B). iScore with both MTT and eRocchio (Featureset D) performs similarly as iScore with eRocchio (Featureset C).

Figure 6.14 shows the current cumulative FMeasure of the classifiers as documents are processed. The figure clearly indicates that eRocchio performs better than all the other classifiers on average, including iScore in its different configurations. However, in Figure 6.15, which shows the FMeasure performance of the classifiers over the 5,000 most recent documents, eRocchio performs better than any of the baseline classifiers. Also iScore with both eRocchio and MTT (Featureset D) perform better in the later time periods for the most recent documents it has seen. Additionally, iScore with eRocchio (Featureset C) outperforms iScore with MTT (Featureset B) in most of the time periods. The figure indicates that iScore with eRocchio and MTT is much more stable in its recommendation performance since there is a much lower drop-off in performance in time periods 3 and 4 than that of the other classifiers, such as eRocchio.
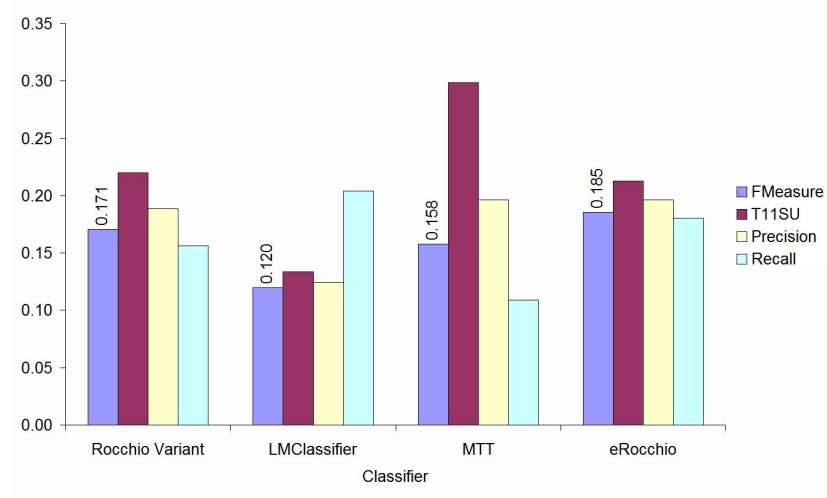
(a) Rocchio variant, LMClassifier, MTT, and eRocchio. eRocchio attains an 8% higher FMeasure score than the Rocchio variant.
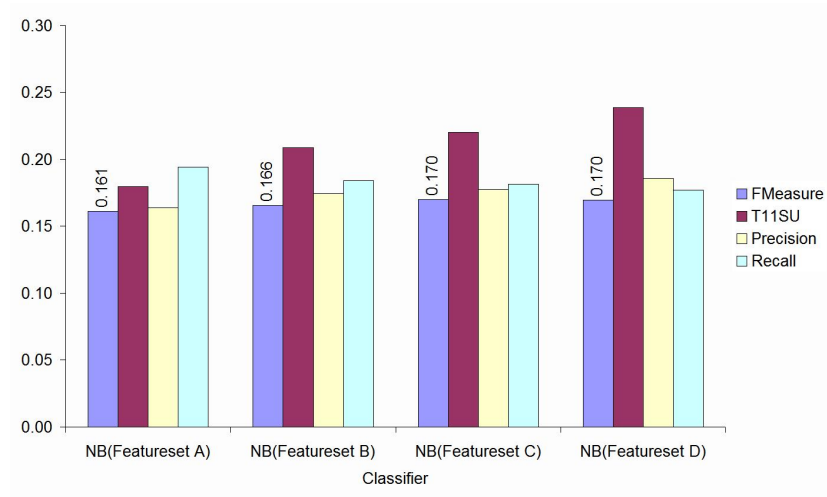
(b) eRocchio and iScore with different featuresets. When eRocchio (Featureset B) is included in the iScore featureset, iScore's FMeasure performance improves by 5.6%, but is still lower than that of eRocchio alone.

Figure 6.12: Average performance for the Rocchio variant, LMClassifier, MTT, eRocchio, and iScore with different featuresets on the Digg dataset.
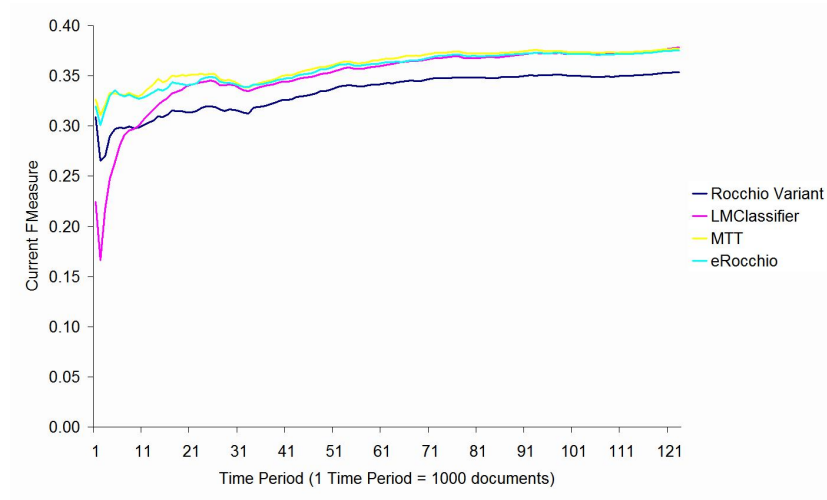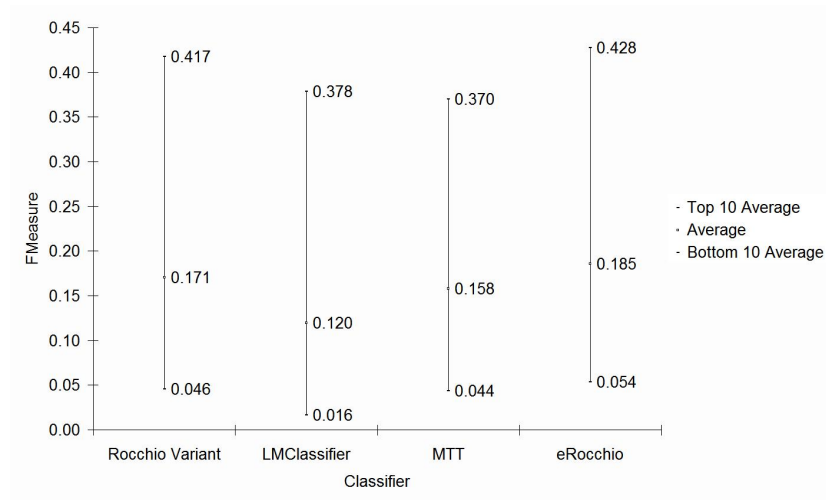
(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
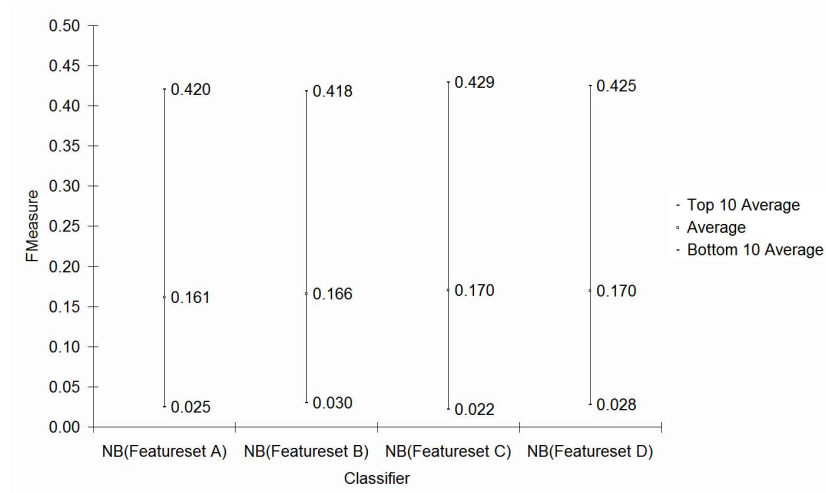
(b) eRocchio and iScore with different featuresets. The addition of eRocchio (Featureset C) beats iScore with MTT (Featureset B).

Figure 6.13: Bottom-10, top-10, and complete average FMeasure for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the Digg dataset.
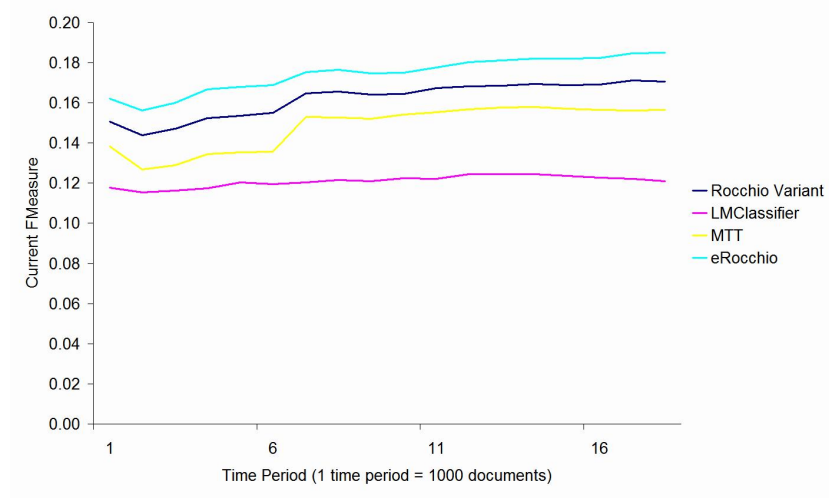
(a) Rocchio variant, LMClassifier, MTT, and eRocchio.
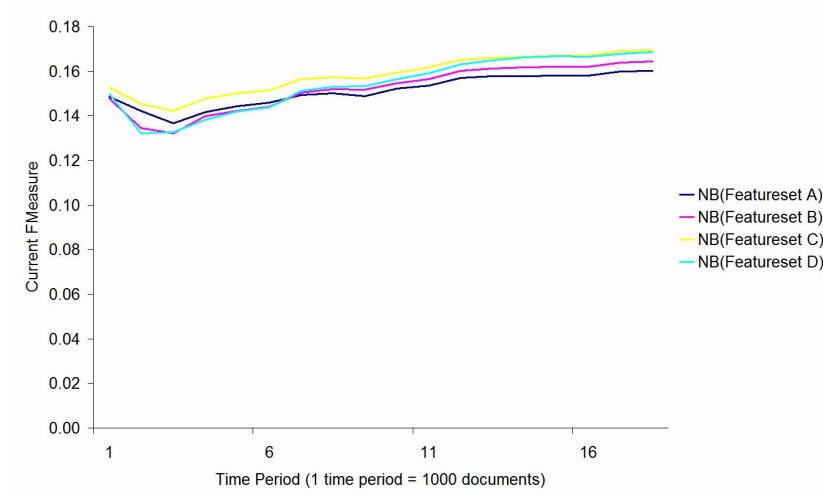
(b) eRocchio and iScore with different featuresets. iScore with eRocchio (Featureset C) outperforms iScore with MTT (Featureset B) in most of the time periods.
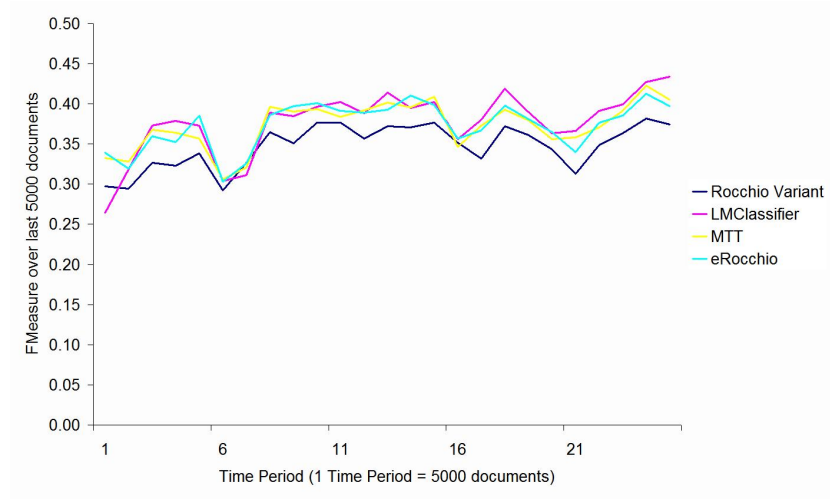
Figure 6.14: Current cumulative FMeasure at specific periods for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the Digg dataset.
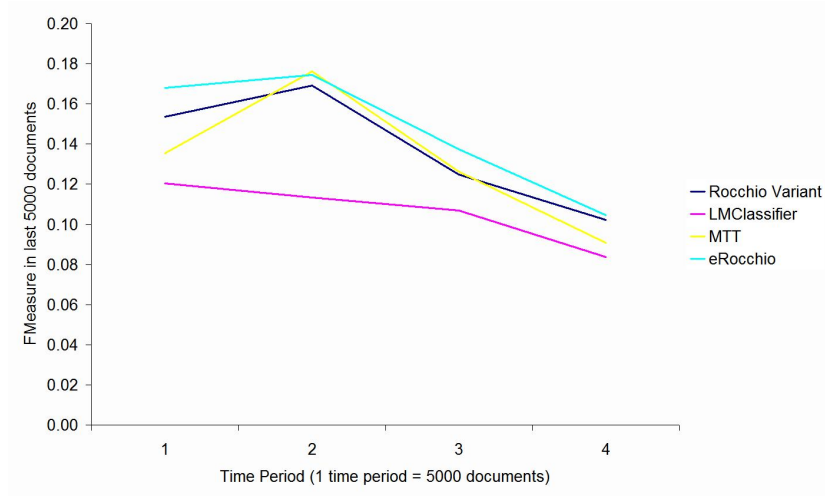
(a) Rocchio variant, LMClassifier, MTT, and eRocchio. eRocchio performs better than any of the baseline classifiers.
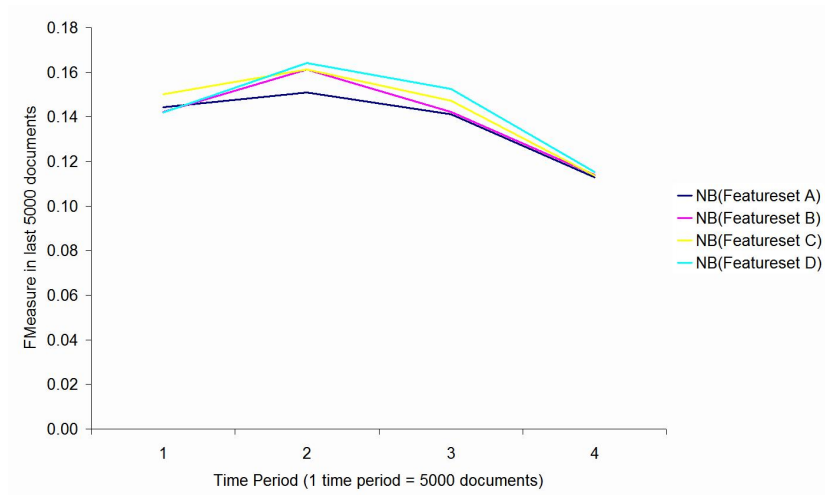
(b) eRocchio and iScore with different featuresets. iScore with both eRocchio and MTT (Featureset D) perform better in the later time periods for the most recent documents it has seen.

Figure 6.15: FMeasure for the 5,000 most recent documents for the Rocchio variant, LMClassifier, MTT, eRocchio, iScore on the Digg dataset.

6.4.5 TREC Adaptive Filter

Although the TREC11 adaptive filter task is to retrieve all articles relevant to a query, regardless of its interestingness to a user, it would be interestig to see how well eRocchio performs 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 6.17. 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 [XYW02]. Also eRocchio is augmented to handle both soft and hard negative articles as shown in Figure 10.1. In the other three datasets, there is no (or little for the tagger dataset) distinction between hard and soft negative articles, as most negative labels were considered soft due to the lack of negative user taggings. Consequently, instead of learning a parameter configuration consisting of only one parameter, in this set of experiments, eRocchio 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 6.17 shows eRocchio outperforming the best classifier from TREC11, ICTAdaFT11Ub, by a significant 4.3 FMeasure points (10% improvement). ICTAdaFT11Ub is the same algorithm as the Rocchio variant in the previous 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
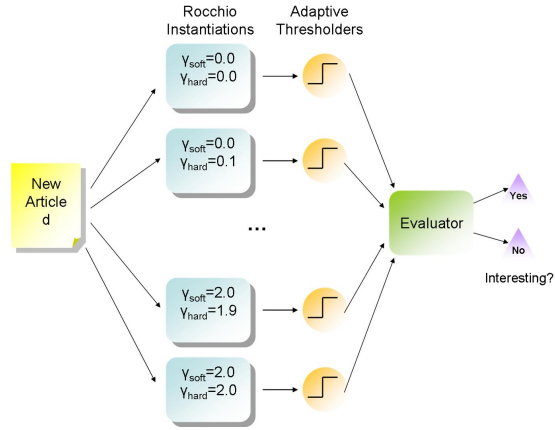
Figure 6.16: eRocchio pipeline with weights for soft and hard negatively-labeled articles.

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. Figure 6.18 shows that eRocchio outperforms ICTAdaFT11Ub in almost every time period and outperforms MTT in every time period.

6.5   Discussion and summary

The optimal learning behavior for a classifier is shown to vary 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 dataset and user behavior. Because of the effectiveness, simplicity, and adaptability of eRocchio, it can replace algorithms such as Rocchio and MTT in iScore.
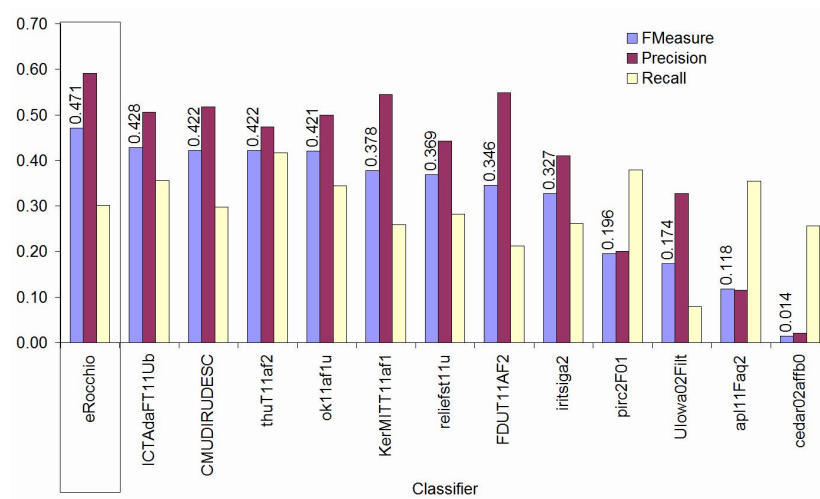
Figure 6.17: Average performance of the top performing filters and eRocchio on the TREC Adaptive Filter task. eRocchio outperforms the best classifier from TREC11, ICTAdaFT11Ub, by 10%.
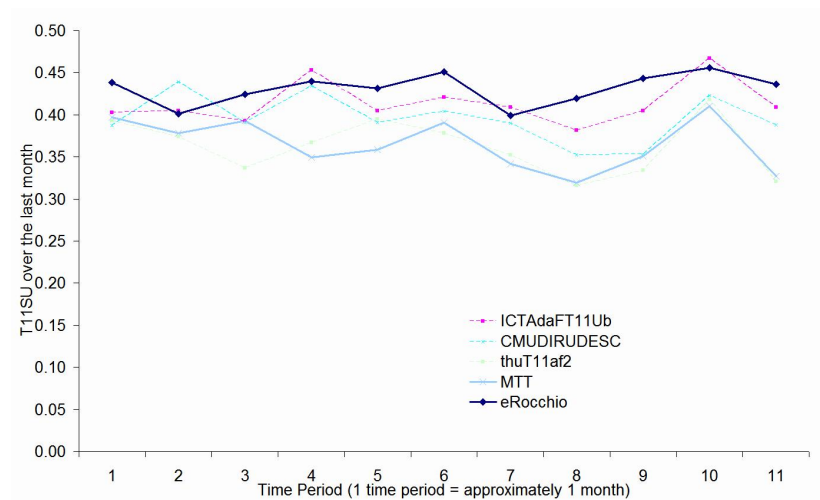
Figure 6.18: Performance for the last month's documents of the top performing filters, MTT, and eRocchio on the TREC Adaptive Filter task.

99

Online learning of parameter configurations can yield better news recommendation results. By itself, eRocchio performs as well as or better than the best baseline classifiers without any prior parameter tuning, for all datasets. When included with iScore along with MTT, eRocchio improves performance by 0.4% and 5.6% in the Yahoo! News and Digg datasets, respectively, while showing a performance decrease only in the tagger dataset. By adapting the online parameter selection algorithm in eRocchio, it can yield 10% better results in the TREC11 Adaptive Filter Task than best performing filter from that 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 extensive fine-tuning.

CHAPTER 7

Additional iScore Features

In addition to the iScore features detailed in the previous chapters, the following features are introduced to address some of the shortcomings of the original features. These features demonstrate improved recommendation performance compared to the original iScore features.

7.1 Language models for interestingness and uniqueness

In Chapter 4, language modeling classifiers are trained and rebuilt in batches to reduce the consumption of computational resources but result in training delay. Language modeling classifiers make use of models that assign a probability to a sequence of tokens. To address this problem, instead of building a classifier on the complete body of all articles (interesting or not) that returns a probability of whether the article is interesting, language models are built on only the title or first paragraph of interesting articles and measure the probability of the text being "generated" from the language model using Equation 4.5, or the probability of the sequence of tokens in the article existing given what has been seen before. In news articles, the title and the first pargraph of an article are often very good summaries of the complete article. So by using a much smaller training text to build the models, the models are more easily updateable. But instead of computing a very large product, which can approach 0 as the number of factors

to be multiplied increase, the log-probability is taken instead, transforming the product of probabilities to a sum of probabilities:

$$f_{LM}(d) = log(P(d)) = \sum_{i=1}^{N} log(P(g_i|g_{i-n+1}, \ldots, g_{i-1})) \qquad (7.1)$$

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

In addition to measuring the probability of an opening paragraph (or title) being "generated" from a language model of interesting first paragraphs (or titles), the probability of the same text being "generated" from a language model of all seen first paragraphs (or titles) is measured. This will serve as another measurement of uniqueness of an article.

7.2 Phrase interestingness

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

> The black dog jumped over the fence.

the following tri-grams would be examined: "The black dog," "black dog jumped," "dog jumped over," "jumped over the," and "over the fence." However, if only noun phrases are looked at, "The black dog" and "the fence" would be the only candidates. Using a noun-phrase extractor provided by [Ope06], noun phrases are

extracted and normalized (making all characters in the phrase lowercase, removing stop-words, and stemming each word in the phrase). The average probability of the interestingness of noun phrases is measured as:

$$f_{PhraseInterestingness}(d) = \frac{\sum_{p \in phrases(d)} \frac{|\text{ times } p \text{ occurs in Int articles}|}{|\text{times } p \text{ occurs in articles}|}}{|phrases(d)|} \qquad (7.2)$$

7.3 Cluster movement

Previous techniques for measuring uniqueness aim at measuring how different an article is compared to previously seen articles. Another method for measuring uniqueness is to measure the impact an article has on its parent topic by measuring how much a cluster of articles has changed when an article is added to it. Articles can be incrementally clustered by using Algorithm 1. In summary, articles are added to a cluster when the TF-IDF vector that represents the centroid of the cluster is highly similar to an article (where $t \geq 0.4$ in the experiments). If there are no highly similar clusters, a new cluster is created using the TF-IDF vector of the article. The clustering algorithm is bootstrapped with a set of clusters to begin with. Cluto [Kar03] is used to generate the initial set of cluster vectors from the small set of Yahoo! News articles. Similarity is defined using the cosine similarity as defined by Equation 4.3.

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

$$\vec{c}_{new} = \vec{c}_{old} + \vec{d} \qquad (7.3)$$

$$f_{ClusterMovement} = \cos(\vec{c}_{new}, \vec{c}_{old}) \qquad (7.4)$$

Input: $\vec{d}$: Document vector, C: Cluster vectors, t: threshold

for $\vec{c} \in C$ **do**

> **if** $\cos(\vec{d}, \vec{c}) \geq t$ **then**
>> $\vec{c} = \vec{c} + \vec{d};$
>
> **end**
>
> **else**
>> $C = C \cup \vec{d};$
>
> **end**

end

<div align="center">Algorithm 1: Clustering.</div>

7.4   Topic-driven freshness

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

cluster that article d is closest to:

$$f_{TDFreshness}(d) = \frac{1}{k} \sum_{d' \in \text{Last } k \text{articles in } topic(d)} Time(d) - Time(d') \qquad (7.5)$$

In the experiments, k is 10 articles, which is a reasonable number of articles. Too large of a number would unfairly weight topics that have existed longer. Too small of a number would yield inaccurate measurements.

7.5 Sliding anomaly detection

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

$$\vec{d}_{summary} = \sum_{d \in \text{Articles from last } k \text{ days}} \vec{d} \qquad (7.6)$$

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

Uniqueness as measured by this measure is defined as follows:

$$f_{SlidingAnomaly}(d) = \cos(\vec{d}_{summary}, \vec{d}) \qquad (7.7)$$

7.6 Experimental results

In these set of experiments, the title and body language models, phrase interestingness, cluster movement, topic-driven freshness, and sliding anomaly detection are evaluated as additional features for iScore with the large Yahoo! News, tagger, and Digg datasets.

7.6.1 Yahoo! News

The first set of experiments evaluate the additional features on the Yahoo! News dataset. Figure 7.1 shows that FMeasure performance improves 0.8 FMeasure points (1.7% improvement) when the features discussed in this chapter (Featureset E) are added to the original iScore features, MTT, and eRocchio (Featureset D). This improvement is mostly due to the 12% increase in recall and a small increase in precision. Closer inspection of the results show that these features improve recommendation performance for both the most difficult and the easiest feeds/users to recommend for. However, most of the improvement is for improving the recommendation performance by 10.5% for the most difficult feeds/users to recommend for.

Figure 7.3 show the cumulative FMeasure of iScore with MTT and eRocchio and iScore with MTT, eRocchio, and the new features as documents are processed. In other words, the figure shows the performance of the classifiers for varying document collection sizes. The figure indicates that the improvement caused by these new features are consistent regardless of the number of documents that have been processed.

Figure 7.4 shows the FMeasure performance of the classifiers for the 5,000 most recent documents. There is improvement caused by these new features for
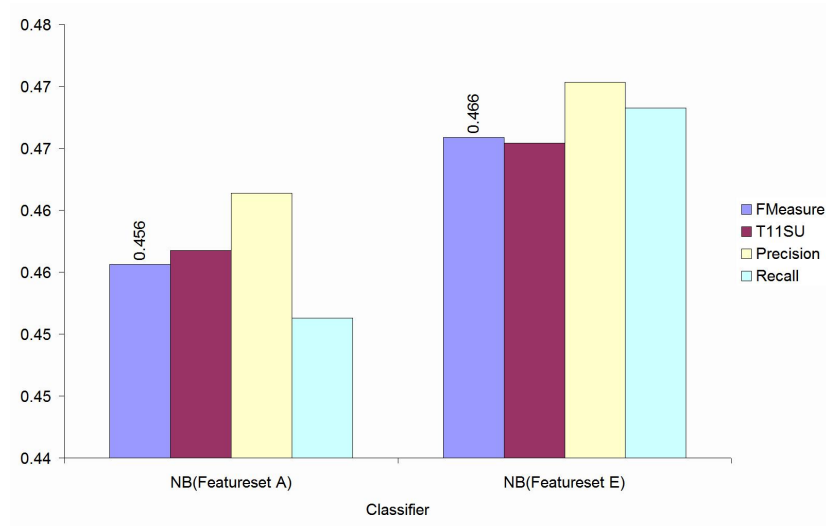
Figure 7.1: Average performance for iScore with the old and new featuresets on the Yahoo! News dataset. Performance improves by 1.7% when the new features are added to the original features, MTT, and eRocchio (Featureset D).

all time periods.

7.6.2 Tagger

The second set of experiments evaluates the additional features on the tagger dataset. Figure 7.5 shows the average FMeasure, T11SU, precision, and recall of iScore with the original featureset, MTT and eRocchio (Featureset D) and iScore with the original featureset, MTT, eRocchio, and the expanded featureset (Featureset E). The figure indicates there is a 14% improvement in FMeasure performance when these new features are added to the featureset of iScore due to increases in precision and recall.

Like the Yahoo! News dataset, Figure 7.6 shows that there is improvement for the easiest user to recommend for. But the performance improvement is
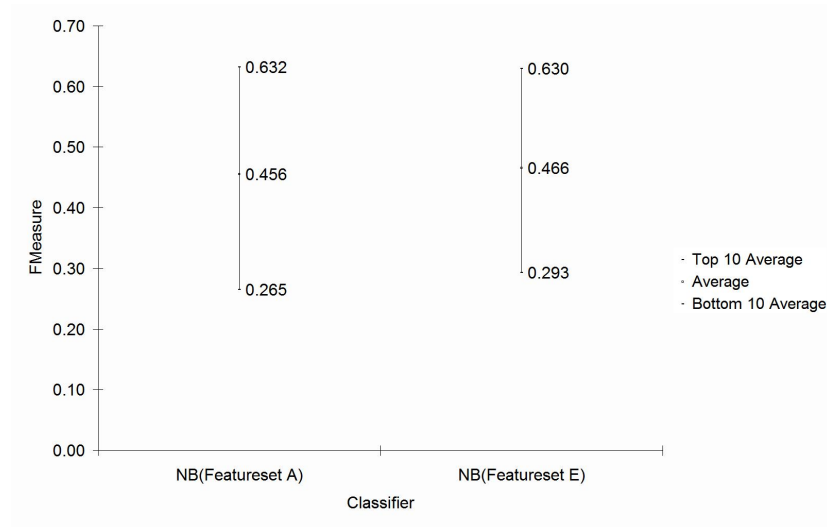
Figure 7.2: Bottom-10, top-10, and complete average FMeasure for iScore with the old and new featuresets on the Yahoo! News dataset. Most of the 5% improvement is for improving the performance for the most difficult feeds/users to recommend for.
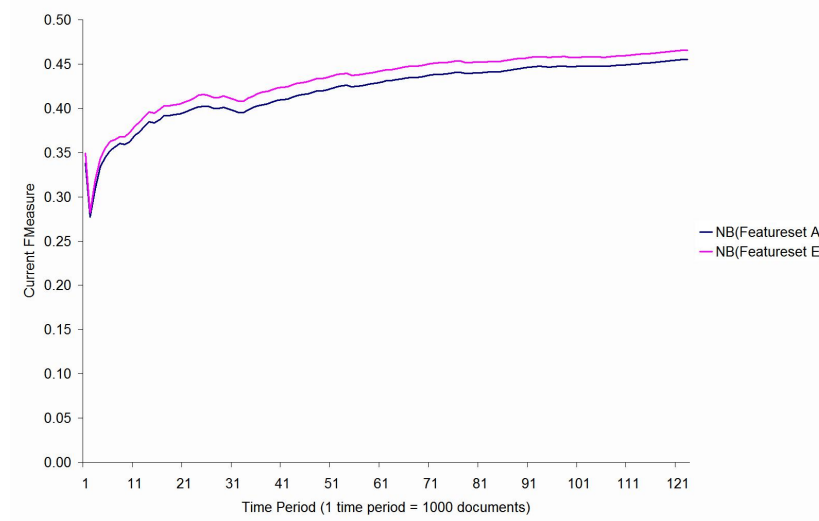
Figure 7.3: Current cumulative FMeasure performance at specific periods for iScore with the old and new featuresets on the Yahoo! News dataset. The improvements caused by the new features are consistent across all time periods.
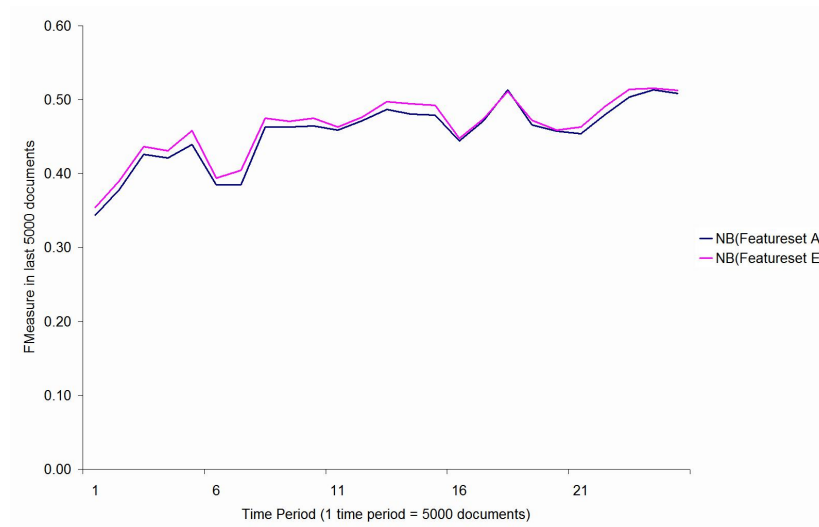
Figure 7.4: FMeasure performance for the 5,000 most recent documents for iScore with the old and new featuresets on the Yahoo! News dataset. There is improvement caused by the new features for all time periods.

109
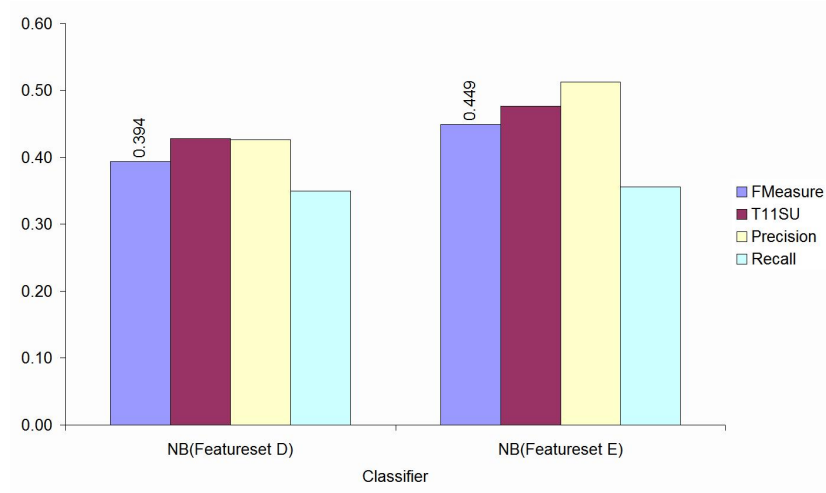
Figure 7.5: Average performance for iScore with the old and new featuresets on the tagger dataset. There is 14% improvement in performance when these new features are added to the featureset of iScore.

much dramatic, with an increase of 158%. Performance of the easiest users with this expanded featureset (Featureset E) are comparable to iScore with MTT, eRocchio, and the original featureset (Featureset D)

7.6.3 Digg

The final set of experiments evaluate the additional features on the Digg dataset. Figure 7.7 shows the average FMeasure, T11SU, precision, and recall of iScore with MTT and eRocchio (Featureset D) and iScore with the MTT, eRocchio, and the additional features discussed in this chapter (Featureset E). The figure indicates that there are improvements in precision, leading to a 2.9% increase in FMeasure and a 4.8% increase in T11SU scores.

Figure 7.8 shows the average FMeasure of all, the worst 10, and the top 10 performing users. The figure indicates that performance improves for all three
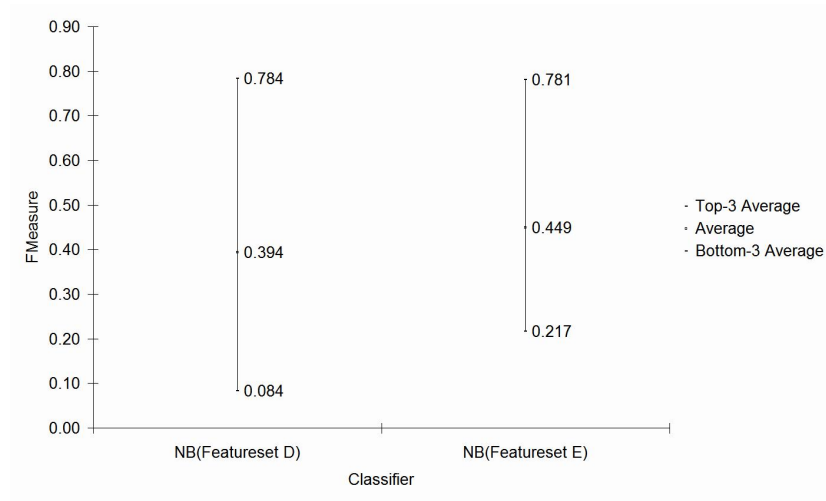
Figure 7.6: Bottom-3, top-3, and complete average FMeasure for iScore with the old and new featuresets on the tagger dataset.
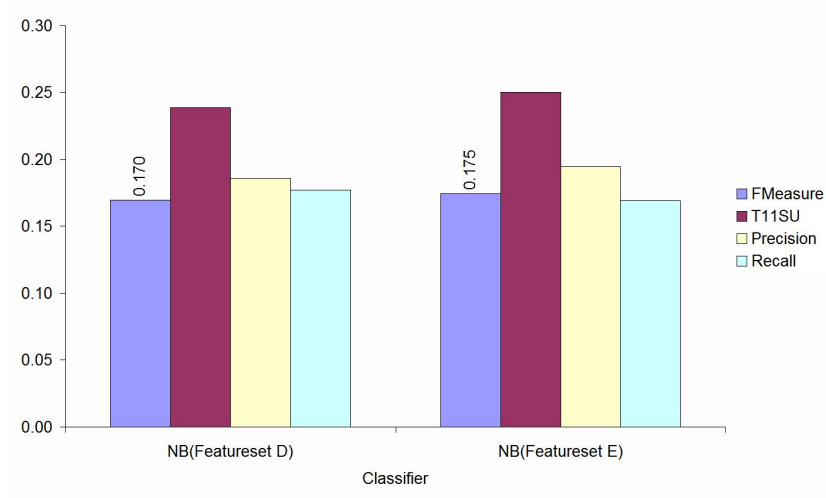
Figure 7.7: Average performance for iScore with the old and new featuresets on the Digg dataset. The addition of the new features leads to a 2.9% increase in FMeasure and a 4.8% increase in T11SU.
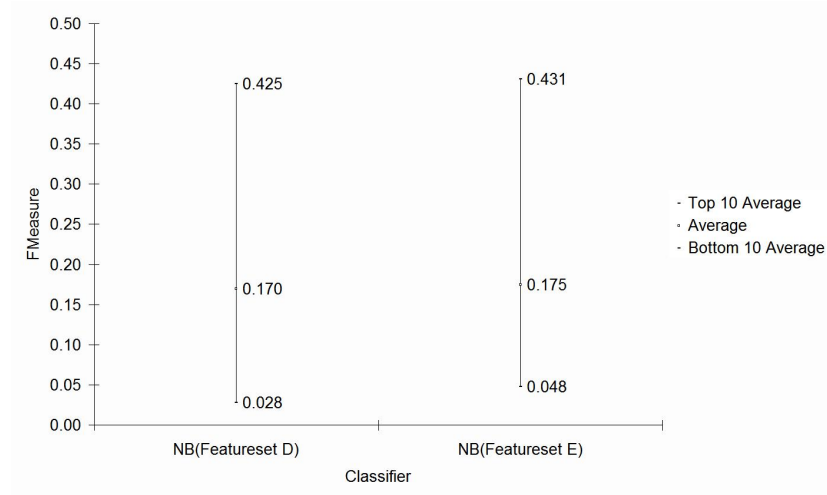
Figure 7.8: Bottom-10, top-10, and complete average FMeasure for iScore with the old and new featuresets on the Digg dataset. Most improvement is for the worst performing users.

categories of users, with the most improvement for the worst performing users.

Figure 7.9 shows the cumulative FMeasure performance of the two featuresets as documents are processed. The figure clearly indicates that there is a margin of improvement with the new additional features.

Figure 7.10 shows the FMeasure performance of the two featuresets over the 5,000 most recent documents processed in each time period. The figure shows that the majority of iScore's average improvement with the expanded featureset is found in the early time periods.

7.7   Discussion and summary

The features discussed in this chapter address some of the deficiencies of some of the other features introduced earlier. Some of the features are new views on older
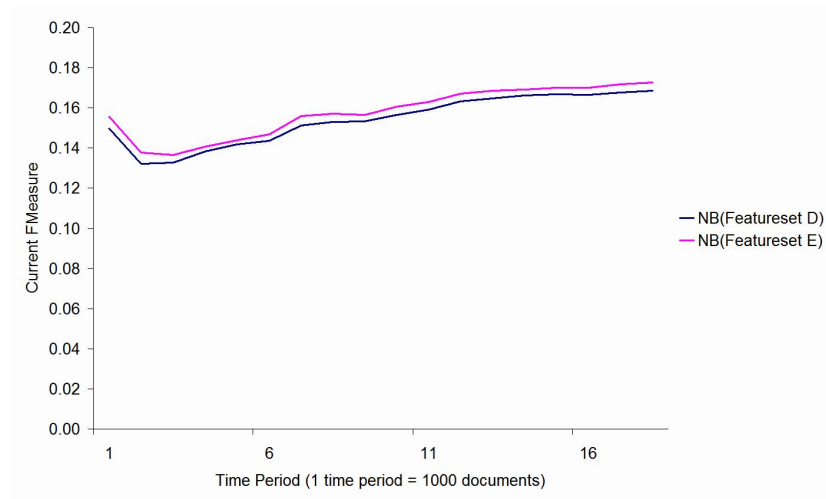
Figure 7.9: Current cumulative FMeasure at specific periods for iScore with the old and new featuresets on the Digg dataset.
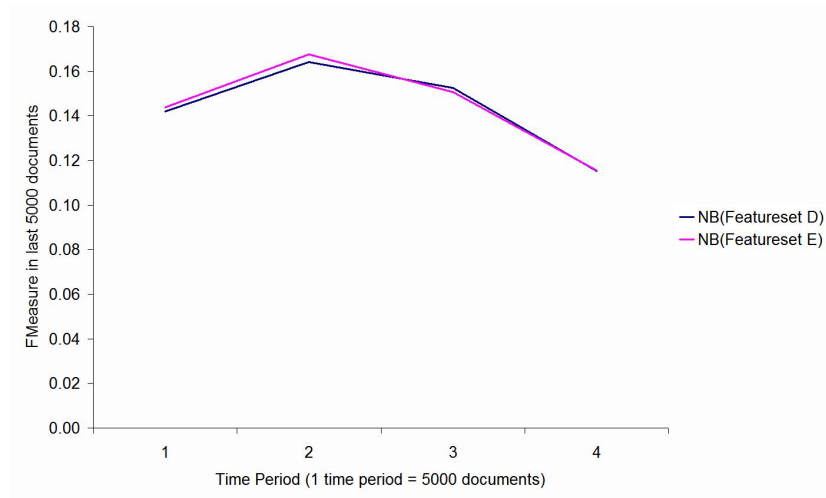
Figure 7.10: FMeasure for the 5,000 most recent documents for iScore with the old and new featuresets on the Digg dataset. The majority of the improvement with the expanded featureset is found in the early time periods.

concepts and are better measurements of a feature. Some features introduce new information not previously used, such as the titles or articles. The addition of these features has improved iScore performance by 1.7%, 6%, and 2.8% in the Yahoo! News, tagger, and Digg datasets, respectively

CHAPTER 8

Online Feature Selection for Interestingness

As stated earlier before, the definition of interestingness varies from user to user. For example, the writing style of an article may be important for one user; whereas, for another user it may be unimportant. As a result, it is not possible to predict which features are important for a specific user before constructing the system and so all features are included for classification. Thus, classification performance suffers initially and requires a significant amount of training to adapt to the presence of useless features. iScore in [PCB07a] suffers from this problem. And the definition of interestingness may even change for a single user over time. For example, the writing style of an article may not be important initially but may evolve to become important later on. The traditional classifiers used by iScore, such as naïve Bayes, can learn to adapt to the changing utility of features, but only with sufficient training. And because of the required long initial training period, the usefulness of the recommendation system suffers. Users of recommendation systems are less inclined to use a system if it requires a significant amount of training before it begins to give accurate recommendations.

To address these problems, online feature selection for naïve Bayes [PCB08a] is introduced. Correlation is used to determine the utility of each feature and take advantage of the conditional independence assumption used by naïve Bayes for online feature selection and classification. The following contributions are made in this area:
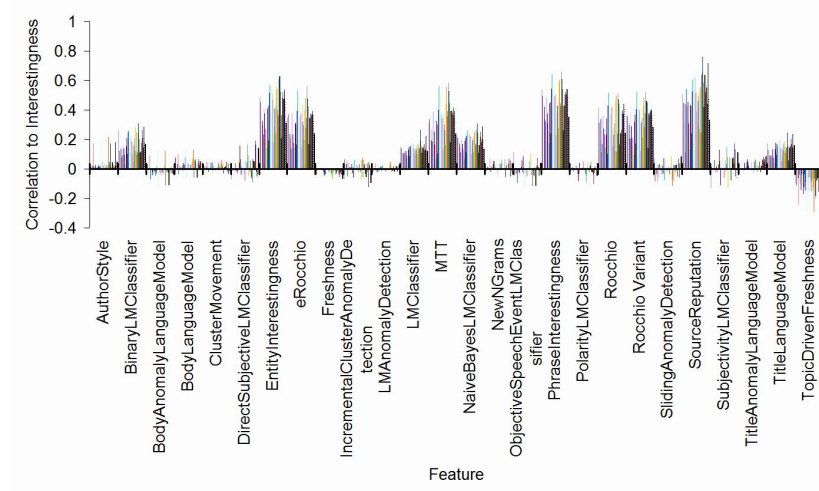
Figure 8.1: Feature correlation with interestingness for Yahoo! News. Each color represents a different proxy user/RSS feed.

1. Augmenting naïve Bayes with online feature selection allows for the fast identification of useless features, improving iScore's initial performance

2. The continual learning of statistics about each feature allows for the invocation of any feature at any time if it has been determined to be useful, addressing the problem of the evolving definition of interestingness

3. By only considering the top-k useful features, evaluation of all possible subsets of features is avoided, making this feature selection approach tractable in an online environment.

8.1   Correlation

The usefulness of features for determining the interestingness of articles are evaluated in [PCB07a]. The Pearsons correlation is used to evaluate the usefuless of
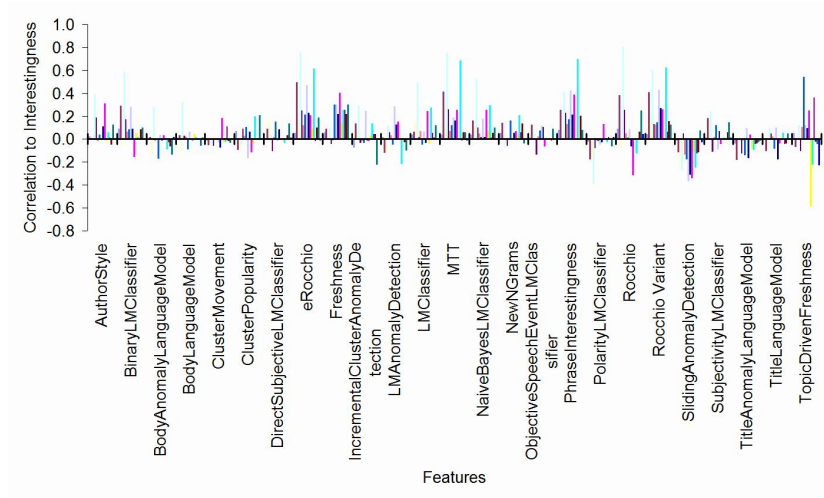
Figure 8.2: Feature correlation with relevancy in the tagger dataset. Each color represents a different user.
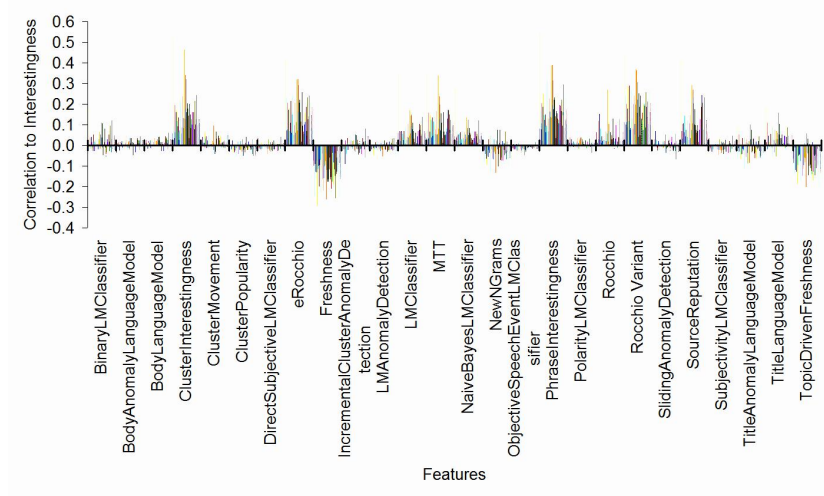
Figure 8.3: Feature correlation with relevancy in the Digg dataset. Each color represents a different user.

117

features. Correlation is defined as:

$$correlation = \frac{E((X - \mu_X)(Y - \mu_Y))}{\sigma_X \sigma_Y} \qquad (8.1)$$

E is the expected value operator. μ_X and σ_X are the average and standard deviaton of the random variable X, respectively.

The features are evaluated using the large collection of news articles from all the Yahoo! News RSS feeds [Yah07]. Figure 8.1 shows the Pearsons correlation of the features (from [PCB07a]) with interestingness in each of the RSS feeds. For most feeds, the topic relevancy and source reputation features are features that have significant direct correlations. 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. A variety of criterion that users may use when evaluating the interestingness of an article are shown. Figures 8.2 and 8.3 show diversity in the features' correlation with interestingness among users as well in the tagger and Digg datasets, respectively.

It is important to note that correlation is not necessarily the best metric for measuring the utility of a feature in document classification since the actual usefulness of a feature can not be determined by studying a single feature in isolation. There are certainly cases where two features that are useless by themselves can be useful when combined together [GE03]. However, correlation is a useful guide if the features were designed to be directly or indirectly correlated with interestingness in mind, as they were for the iScore features. And by coupling this independent correlation metric with a classifier that assumes that each feature is independent, such as naïve Bayes, performance of the classifier should improve. In [GE03], information gain and correlation are suggested for feature ranking. Information gain is difficult to compute in an online fashion because the appropriate discretization is difficult to determine if the entire data is not avail-

118

able during evaluation (as in an online streaming environment). Consequently, correlation is used instead due to its simple online computability and its lack of a need for discretization.

8.2   Online feature selection with naïve Bayes

Based on Bayes' theorem, a naïve Bayes classifier is a simple and fast probabilistic classifier that assumes that features are conditionally independent [WF05]. In the context of classifying articles, the probability of an article being interesting is defined by a naïve Bayes classifier as:

$$P(Int|f_1,\ldots,f_n) = \frac{1}{Z}P(Int)\prod_{i=1}^{n}P(f_i|Int) \tag{8.2}$$

where Z is a scaling factor dependent on $f_1,\ldots,f_n$, and Int is the interesting article class. The probability $P(f_i|Int)$ is estimated using kernel estimators [JL95]. During classification, when a feature is unavailable, it is simply ignored, which is equivalent to marginalizing over them.

A naïve Bayes classifier is ideal for online classification since the statistics necessary for computing the probabilities are incrementally updateable. Additionally, a naïve Bayes classifier allows for the exclusion of features during classification so subsets of features can be used for classification while all features can be used for training. Any other incremental classifier that allows for the exclusion of features during classification could be used. But for the purposes of this study, a naïve Bayes classifier, a classifier that has been known to be highly accurate, is used to evaluate the online feature selection approach proposed here.

Ideally, an article is classified using only the most useful features for a specific user. Thus, given a set of n features, the features are ordered by their current absolute Pearsons correlation to interestingness. The top-k most highly correlated

features for classification, where $k = 1, \ldots, n$, are considered as subset candidates. Thus, for every document, n classification scores (each referred to as a subset score) are generated; one score for each subset. The overall score is the subset score associated with the subset of features with the highest FMeasure statistic. Because of the conditional independence of the features, only a single set of statistics is needed to be maintained, (in the form of kernel estimators) related to $P(f_i|Int)$ and $P(f_i)$ even though n classification scores is generated for each document. For a subset of features of size less than n, features not in the subset are essentially ignored when generating a classification score from the naïve Bayes classifier.

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

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

Since only subsets of features with the highest correlations are considered for each document, as opposed to all possible subsets, this feature selection solution is tractable. Sets consisting of only features with low correlation with interestingness would be expected to be very low performing for document classification; whereas, sets of features with high correlation would be expected to be higher performing. Because only the top-k most highly correlated features are considered, subsets consisting of only low correlated features are never considered. And

Input: d: Document, f: Features, T: Thresholders, *Tagging*: Tagging, C:
Classifier, S: Set of feature sets, M: Set of FMeasure statistics

Output: Interestingness of article

if S *is empty* **then**
 | $I(d) = T_{all}(C(f_1(d), f_2(d), \ldots, f_n(d)))$;
end

else
  $maxF = 0$, $s = \emptyset$;

  Sort all f by its absolute Pearson's correlation;

  for $i = 1$ *to* n **do**
    $s = s \cup f_i$;

    if $maxF < M(s)$ **then**
      $maxF = M(s)$;

      $I(d) = T_s(C(s))$;
    end
  end
end

Update(d, f, T, *Tagging*, C, S, M);

return $I(d)$;

Algorithm 2: Online Feature Selection and Classification

from document to document, one would expect to see very similar top-k subsets and so it may be sufficient to only update the FMeasure statistics for each top-k subsets considered for that document. The online feature selection and classification is outlined in Algorithm 2. The details regarding the updates to the classifier, thresholders, and statistics are detailed in Algorithm 3.

For example in Figure 8.4, let there be three features that are ordered in terms of correlation with interestingness: f_1, f_2, and f_3. Using this order, the following subsets are evaluated for their classification effectiveness: $\{f_1\}$, $\{f_1, f_2\}$, and

Input: d: Document, f: Features, T: Thresholders, $Tagging$: Tagging, C:

Classifier, S: Set of feature sets, M: Set of FMeasure statistics

for $i = 1$ to n **do**

$\quad s = s \cup f_i$;

$\quad$ **if** $s \notin S$ **then**

$\quad\quad |\quad S = S \cup s$;

$\quad$ **end**

$\quad M(s) =$ Update the FMeasure of $C(s)$ with $Tagging$;

$\quad$ Update T_s with $Tagging$;

$\quad$ Update Pearson's correlation of f_i to interestingness with $Tagging$;

end

Algorithm 3: Update
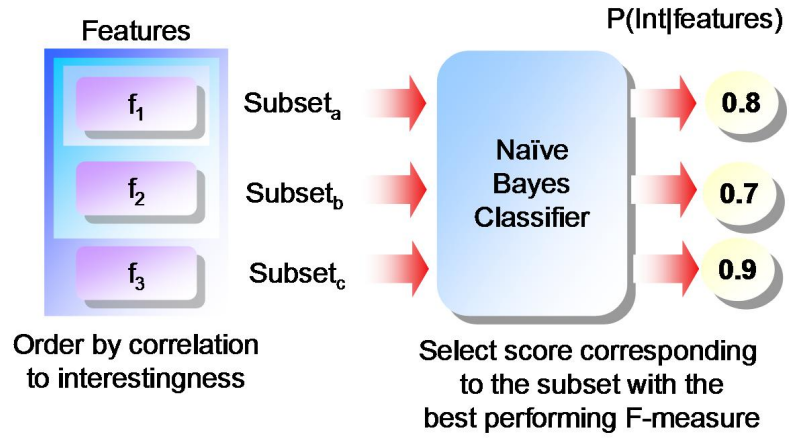
Figure 8.4: Online feature selection with naïve Bayes.

122

$\{f_1, f_2, f_3\}$. If the subset $\{f_1, f_2\}$ has yielded the highest FMeasure performance so far, then it will be used to classify the next document in the document stream. After the real interestingness of the article is revealed to the system, then the classifier is updated along with the correlation of each feature with interestingness and the FMeasure of each subset. The process repeats with the next article in the document stream.

8.3 Experimental results

In these set of experiments, feature-selection naïve Bayes (FSNB) is evaluated with the large Yahoo! News, tagger, and Digg datasets.

8.3.1 Yahoo! News

The first set of experiments evaluate online feature selection on the Yahoo! News dataset. Figure 8.5 shows the average FMeasure, T11SU, precision, and recall for iScore with the original featureset (Featureset A) and iScore with the expanded featureset that includes the original featureset, MTT, eRocchio, and the additional features discussed in Chapter 7 (Featureset E). Also shown are iScore with naïve Bayes (NB) as its overall classifier and features-selection naïve Bayes (FSNB) as its overall classifier. The figure indicates that FMeasure improves by 0.9 FMeasure points for Featureset A and 0.4 FMeasure points for Featureset E. The performance increase is due to dramatic improvements in precision despite some drops in recall, resulting in a higher FMeasure that weights precision twice as much as recall.

Figure 8.6 shows the average FMeasure of all, the most difficult, and the easiest feeds/users to recommend for. For Featureset A, improvement is found
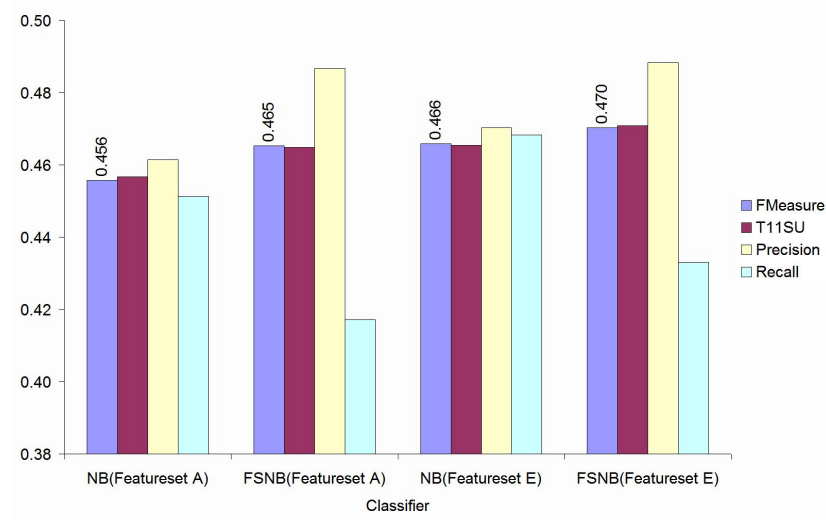
Figure 8.5: Average performance for iScore with/without feature selection working on the Yahoo! News dataset. FMeasure improves by 0.9 FMeasure points for Featureset A and 0.4 FMeasure points for Featureset E.

for both the bottom 10 and the top 10 feeds. For Featureset B, performance improves only for the top 10 feeds.

Figure 8.7 shows the FMeasure performance of iScore with naïve Bayes and feature-selection naïve Bayes as the overall classifier as documents are processed. The figure indicates that for both featuresets, feature-selection naïve Bayes outperforms naïve Bayes for all time periods, with better performance improvements for Featureset A.

Figure 8.8 shows the FMeasure performance of iScore with naïve Bayes and feature-selection naïve Bayes as the overall classifier over the past 5,000 document seen. This figure, like the previous one, shows that online feature selection improves recommendation performance for both featuresets, with better performance improvements for Featureset A. The figure also shows the most improve-
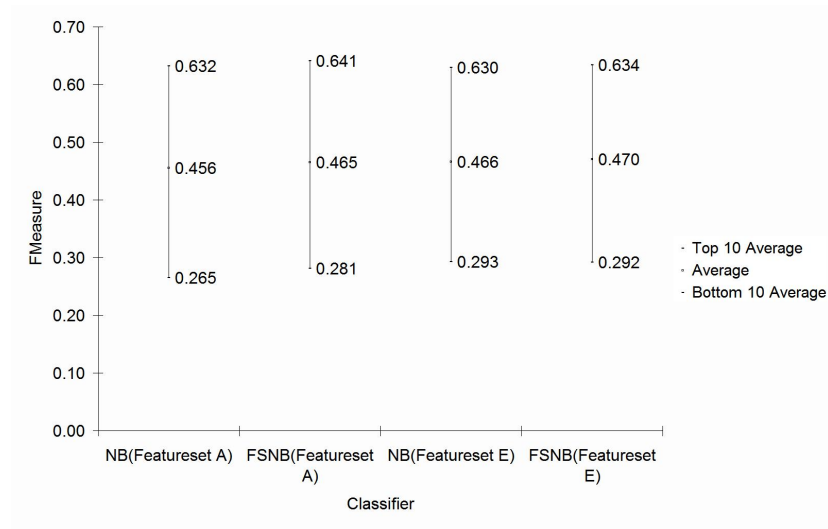
Figure 8.6: Bottom-10, top-10, and complete average FMeasure for iScore with/without feature selection on the Yahoo! News dataset. There is improvement for the bottom 10 feeds for both featuresets.
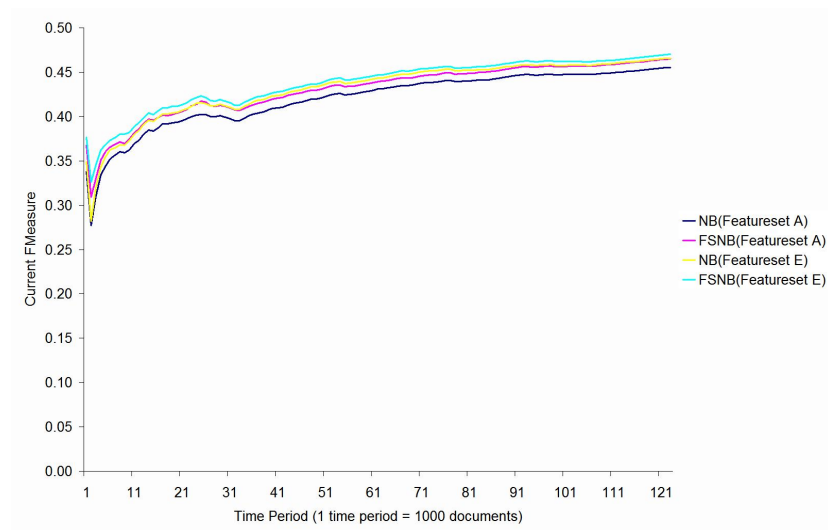
Figure 8.7: Current cumulative FMeasure performance at specific periods for iScore with/without feature selection on the Yahoo! dataset.
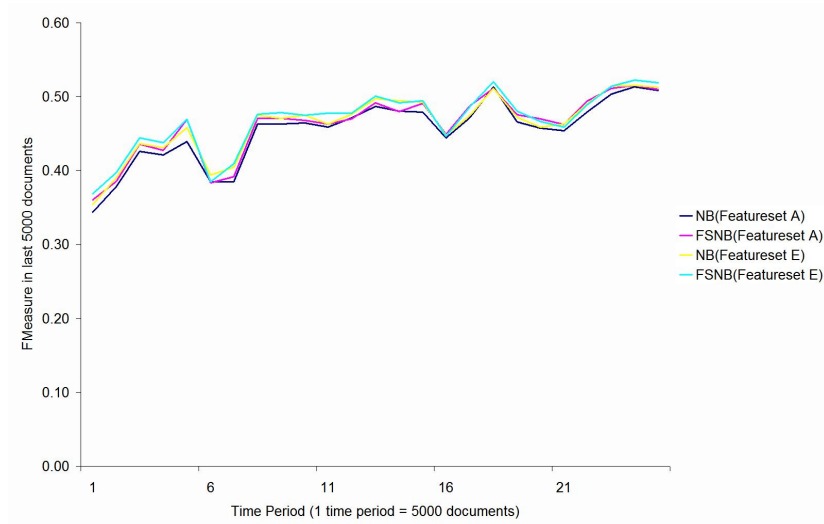
Figure 8.8: FMeasure performance for the 5,000 most recent documents for iScore with/without feature selection on the Yahoo! News dataset. The most improvement is found in the early time periods.

ment in the early time periods.

8.3.2 Tagger

The second set of experiments evaluate online feature selection with the tagger dataset. Figure 8.9 shows the average FMeasure, T11SU, precision, and recall of naïve Bayes and feature-selection naïve Bayes. like the Yahoo! News dataset, feature-selection naïve Bayes performs better than naïve Bayes for both featuresets, with an improvement of 18.9% and 11.1% for the Featuresets A and E, respectively.

Figure 8.10 shows the average FMeasure of all, the most difficult, and the easiest user. Like the previous figure, feature-selection naïve Bayes shows increases in performance compared to naïve Bayes. For both featuresets, performance im-

126
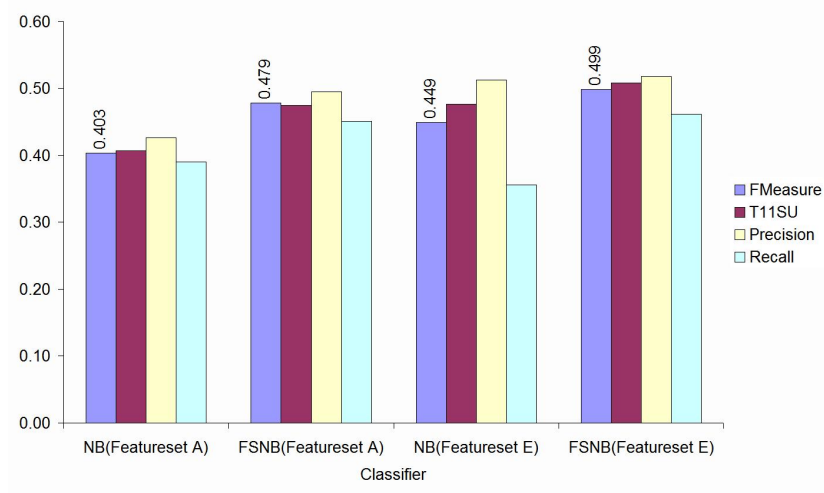
Figure 8.9: Average performance for iScore with/without feature selection on the tagger dataset. Online feature selection improves performance by 18.9% and 11.1% for Featuresets A and B, respectively.

proves for all three groups of users.

8.3.3 Digg

The final set of experiments evaluate online feature selection on the Digg dataset. Figure 8.11 shows the average FMeasure, T11SU, precision, and recall of naïve Bayes and feature-selection naïve Bayes operating over the original featureset and the expanded featureset. The figure indicates that online feature selection improves FMeasure by 3.7% and T11SU by 25.7% for the Featureset A. For Featureset E, FMeasure improves by 3% and T11SU improves by 7%. Increases in performance in both features are due to improvements in precision with a slight decreases in recall. This is consistent with the findings for the Yahoo! News dataset.

Figure 8.12 shows the average FMeasure of all, the 10 most difficult, and
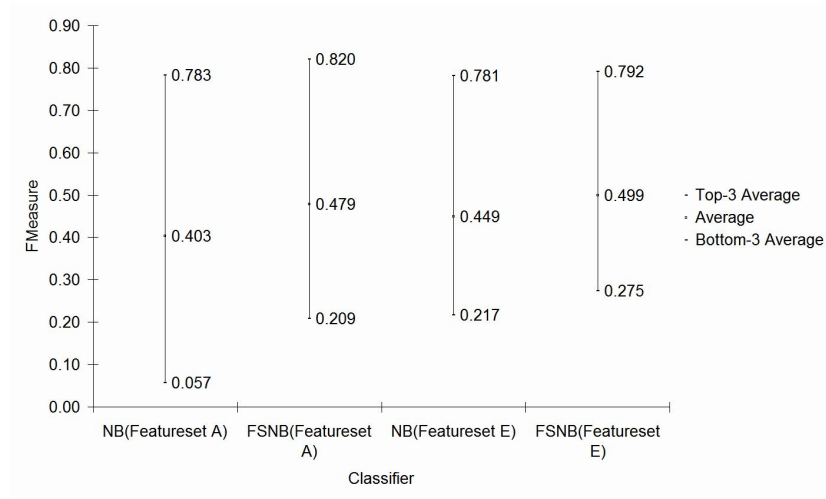
Figure 8.10: Bottom-3, top-3, and complete average FMeasure for iScore with/without feature selection on the tagger dataset.
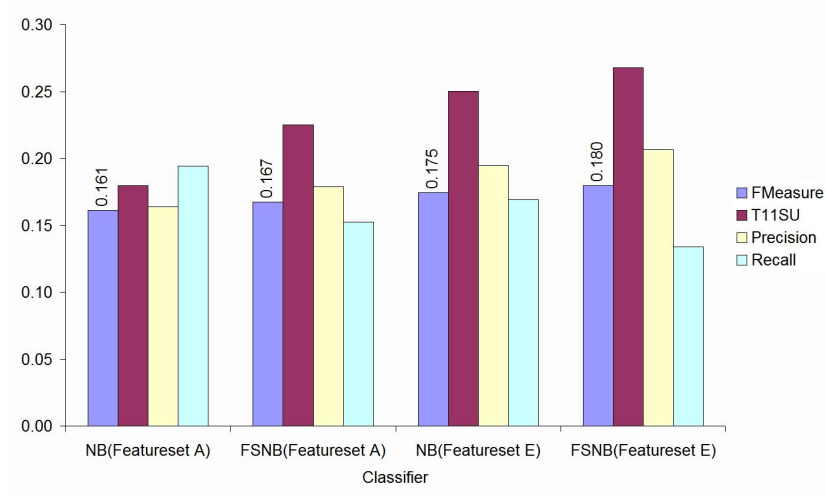
Figure 8.11: Average performance for iScore with/without feature selection on the Digg dataset. Online feature selection improves FMeasure by 3.7% and T11SU by 25.7% for the Featureset A. For Featureset E, FMeasure improves by 3% and T11SU improves by 7%.
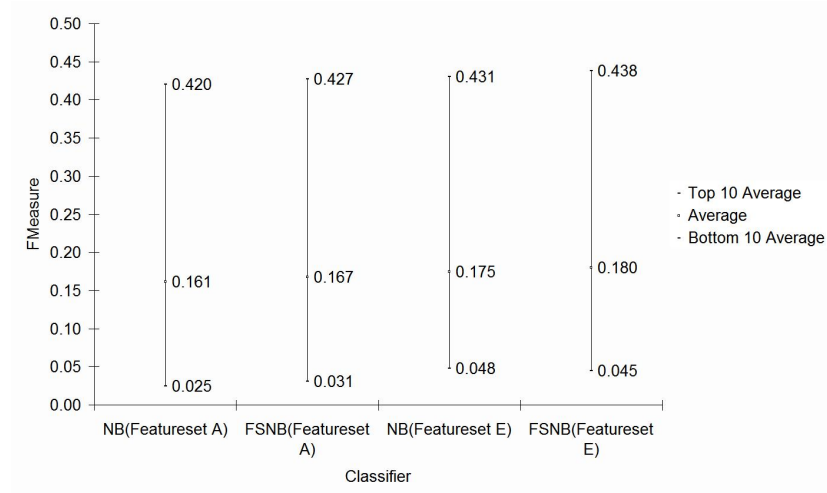
Figure 8.12: Bottom-10, top-10, and complete average FMeasure for iScore with/without feature selection on the Digg dataset.

the 10 easiest users to recommend for. The figure shows that recommendation results improve for the easiest users by employing feature selection. Results for feature selection on the most difficult users are mixed, showing improvement for the original featureset (Featureset A) and showing performance loss for the expanded featureset (Featureset E).

Figure 8.13 shows the cumulative FMeasure of the classifiers as documents are processed. The figure indicates that the performance improvement introduced by online feature selection is consistent for all documents processed. Figure 8.14 shows the FMeasure of the classifiers on the 5,000 most recent documents. The figure shows that there is a significant spike in performance for feature selection naïve Bayes, starting at the first period. Although performance drops for the subsequent periods, the performance of feature selection naïve Bayes remains at or above that of naïve Bayes.
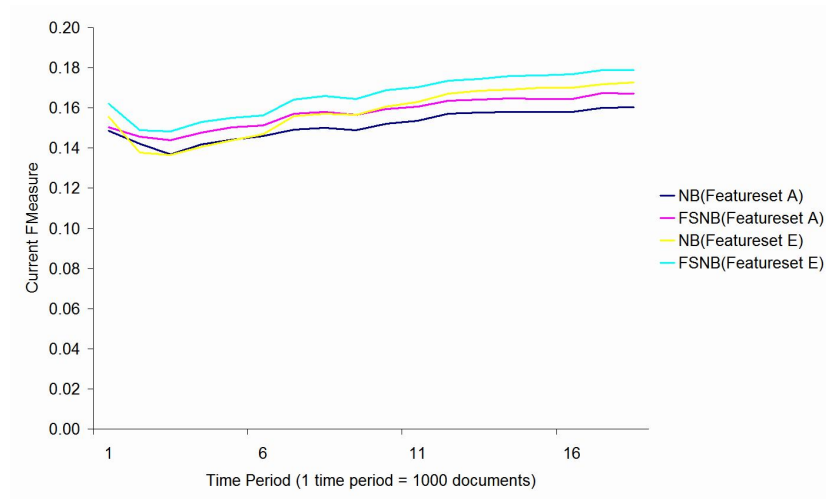
Figure 8.13: Current cumulative FMeasure at specific periods for iScore with/without feature selection on the Digg dataset.
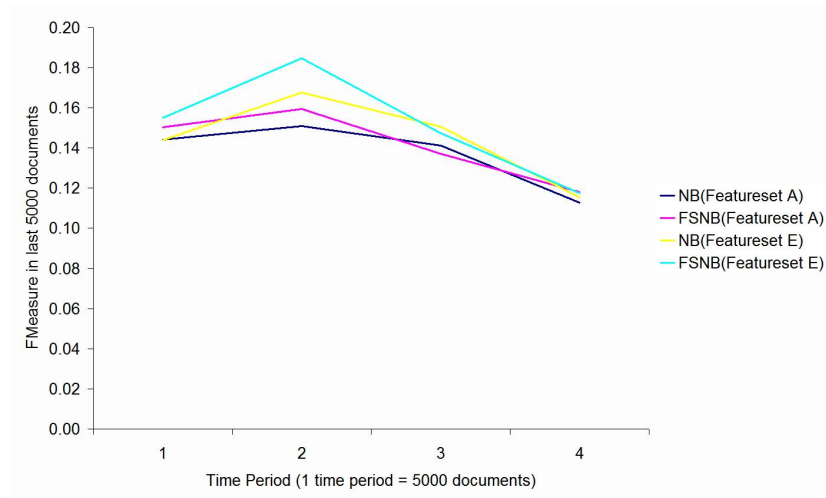
Figure 8.14: FMeasure for the 5,000 most recent documents for iScore with/without feature selection on the Digg dataset. There is a significant spike in performance for feature selection naïve Bayes, starting at the first period

8.4   Discussion and summary

Online feature selection for naïve Bayes significantly improves the accuracy in recommending news articles. By learning which features are useful and useless for identifying interesting articles for a specific user in an online setting, the augmented naïve Bayes can adapt quickly to changes in the definition of what makes an article interesting with little training data. Using correlation, the usefulness of a feature can be determined quicker than the probability distributions used by a simple naïve Bayes classifier. By considering only useful subsets of features, this online feature selection approach is efficient while yielding higher quality results that are better than the traditional naïve Bayes classifier. Online parameter selection has improved recommendation results by 2%, 11.1% and 3.7% in the Yahoo! News, tagger, and Digg datasets, respectively.

CHAPTER 9

Recommendation Results Summary

In this chapter, the final recommendation performance results are summarized for the Yahoo! News, tagger, and Digg datasets, comparing the best performing classifier with the original iScore configuration, the Rocchio Variant (which performed the best in the last TREC Adaptive Filter task), a language modeling classifier, MTT, and eRocchio.

9.1 Yahoo! News

For the Yahoo! News collection, the best performing classifier seen is iScore using feature-selection naïve Bayes as its overall classifier operating on the original featureset, MTT, eRocchio, and the additional features discussed in Chapter 7 (Featureset E).

Figure 9.1 shows the average FMeasure, T11SU, precision, and recall of the Rocchio variant (which best performed in the TREC Adaptive Filter task), the language modeling classfier, MTT, eRocchio, iScore with the original featureset (NB(Featureset A)), and iScore with feature-selection naïve Bayes and the expanded featureset (FSNB(Featureset E)). The figure indicates that 0.47 FMeasure can be achieved with the expanded featureset and online feature selection, due to high precision and high recall. This is 24% better than the best baseline classifiers. And this is 3% better than iScore in its original configuration.
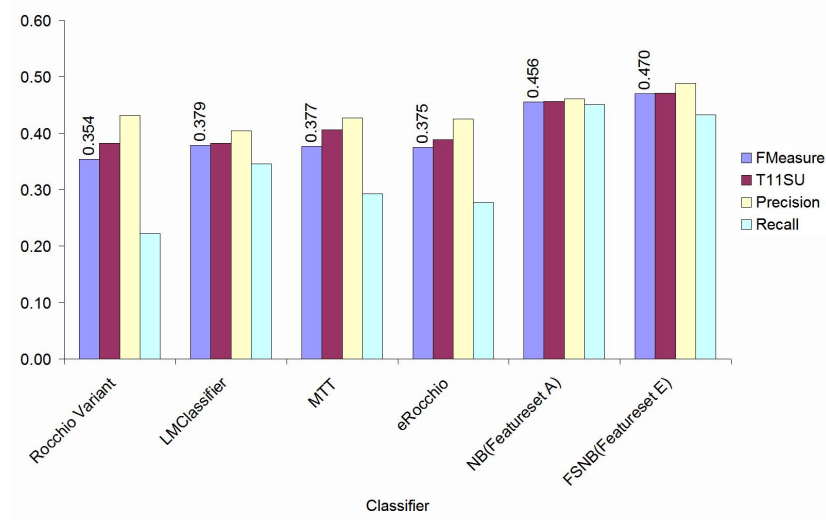
Figure 9.1: Average performance for the Yahoo! News dataset. iScore with feature selection and the expanded featureset is 24% better than the best baseline classifiers.

Figure 9.2 shows the average FMeasure of all, the top 10 and bottom 10 performing feeds/users. The figure shows that iScore with online feature selection and the expanded featureset (i.e., FSNB(Featureset E)) can give much better performance for the worst performing feeds/users than all of the baseline classifiers. It also shows that it has the best performance of the best performing feeds/users as well.

Figure 9.3 shows the cumulative FMeasure of the classifiers as documents are processed. The figure indicates that there is a period of time where iScore must learn and performance gradually improves. The figure also indicates the iScore with online feature selection and the expanded featureset consistently outperforms all the other classifiers.

A similar result is seen in Figure 9.4, which shows the FMeasure performance of the classifiers over the 5,000 most recent documents. The figure also shows
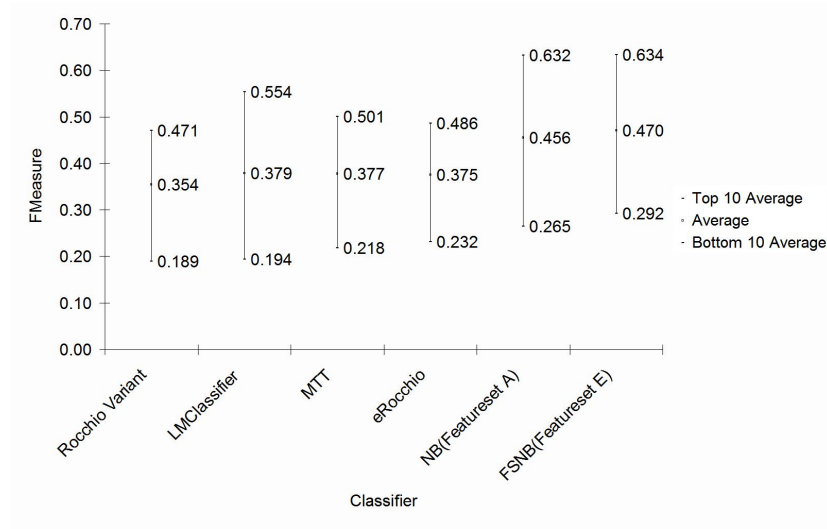
Figure 9.2: Bottom-10, top-10, and complete average FMeasure for the Yahoo! News dataset. iScore with online feature selection and the expanded featureset can give much better performance for the average, worst, and best performing feeds/users than all of the other classifiers.
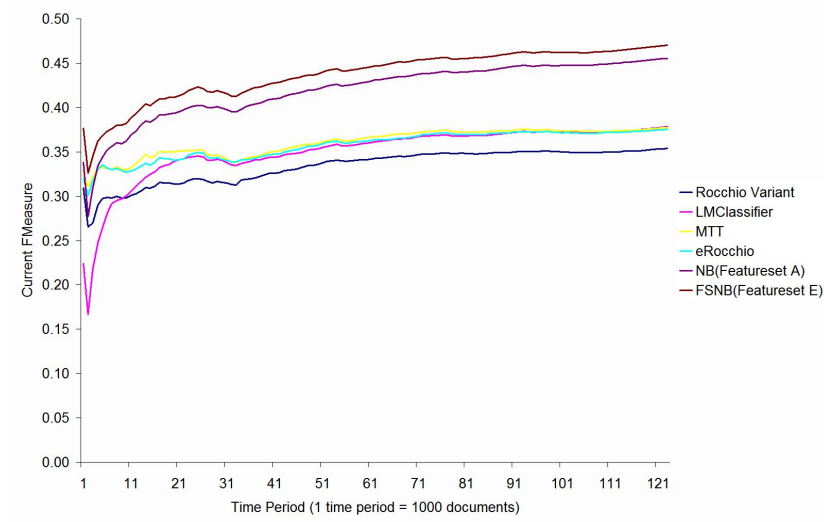
Figure 9.3: Current cumulative FMeasure performance at specific periods for the Yahoo! News dataset. iScore with online feature selection and the expanded featureset consistently outperforms all the other classifiers.

that there is a dramatic drop in performance between periods 6 and 7 for all classifiers. This is most likely due to a long pause in the data collection for this time period.

9.2   Tagger

For the tagger dataset, the iScore with the expanded featureset and onlne feature selection performs the best, much like the Yahoo! News collection. Figure 9.5 shows that very high precision and recall can achieved with iScore with the expanded featureset (Featureset E) with online feature selection over the baseline classifiers and iScore in its original configuration (Featureset A).

Figure 9.6 shows iScore with the expanded featureset (Featureset E) and online feature selection performs better due to recommendation improvements for
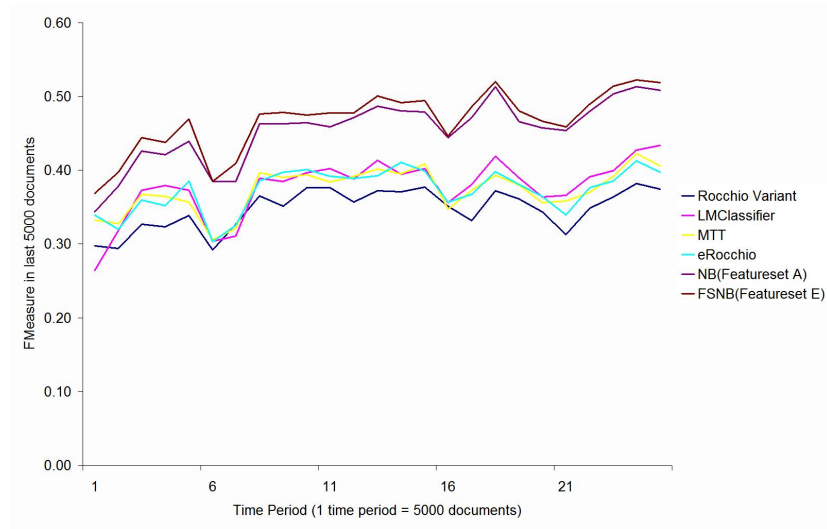
Figure 9.4: FMeasure performance for the 5,000 most recent documents for the Yahoo! News dataset.
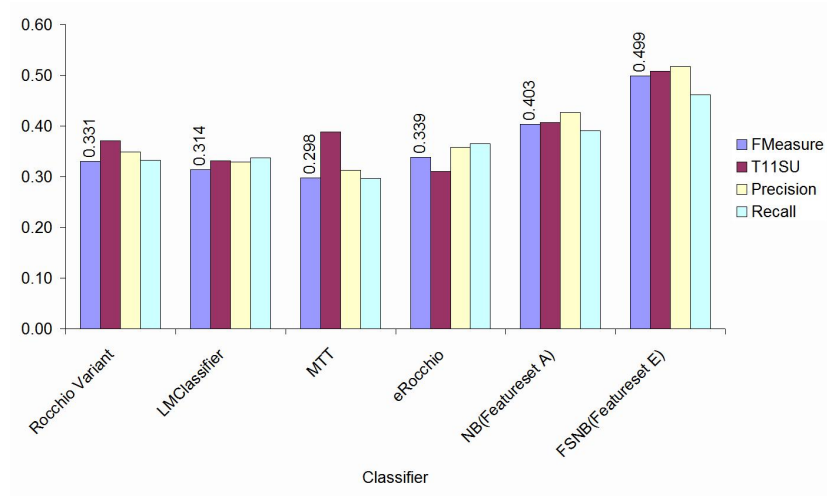
Figure 9.5: Average performance for the tagger dataset. Very high FMeasure can achieved be with iScore with the expanded featureset (Featureset E) over the baseline classifiers and iScore in its original configuration (Featureset A).

136
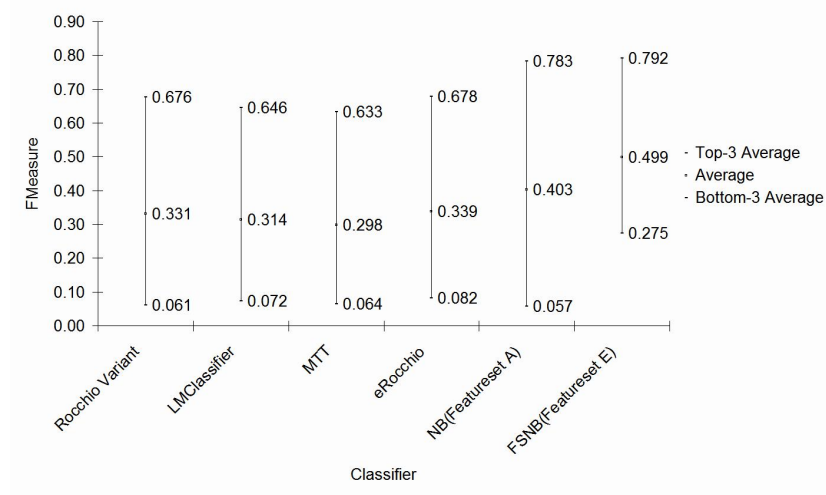
Figure 9.6: Bottom-3, top-3, and complete average FMeasure for the tagger dataset.

the easiest users and most difficult users and for most users in general. The overall performance improvement provided by the expanded featureset and online feature selection over iScore in its original configuration is 23.8%. The improvement over the best baseline classifier (i.e., the Rocchio variant) is 50.7%.

9.3 Digg

For the Digg dataset, the best classifier seen is iScore with online feature selection using the original featureset, MTT, eRocchio, and the additional features discussed in Chapter 7. Figure 9.7 shows the average FMeasure, T11SU, precision, and recall of the classifiers on the Digg data. Although, the average performance results show that eRocchio performs better in terms of FMeasure alone, iScore with feature selection and the expanded featureset (i.e., FSNB(Featureset E)) has a much higher T11SU score and precision than all the other classifiers with

137
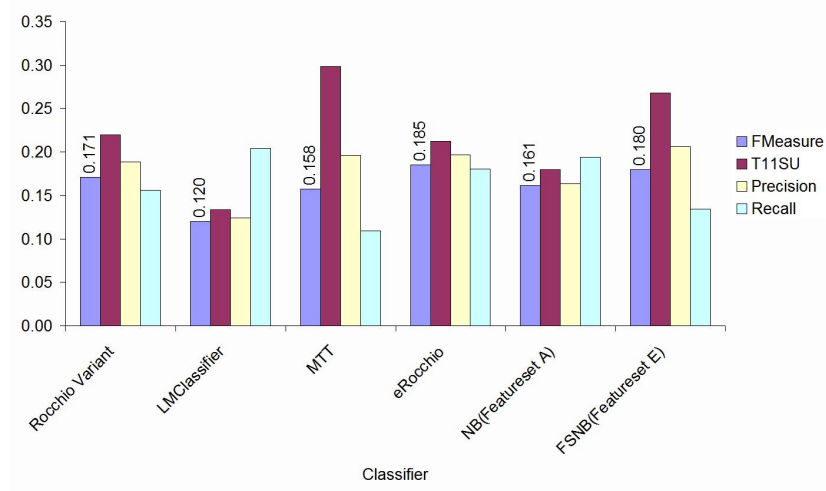
Figure 9.7: Average performance for the Digg dataset. iScore with feature se-
lection and the expanded featureset (FSNB(Featureset E)) has a much higher
T11SU score and precision than all the other classifiers with a high FMeasure
score.

a high FMeasure score, as shown in Figure 9.7. iScore with feature selection and
the expanded featureset is 11.8% better than iScore in its original configuration.
Also, it is 5.2% better than the best baseline classifier (i.e., the Rocchio variant).

Figure 9.8 shows the average FMeasure of all, the easiest, and the most dif-
ficult users to recommend for. The figure indicates that iScore with feature se-
lection and the expanded featureset can recommend news articles for the easiest
users the best. Additionally, this figure shows that new iScore configuration clas-
sifier has improved upon the recommendation performance of the original iScore
configuration.

Figure 9.9 shows the cumulative FMeasure performance of the various clas-
sifiers as documents are processed. The figure suggests that eRocchio performs
better than the other classifiers on average. The figure also shows that iScore
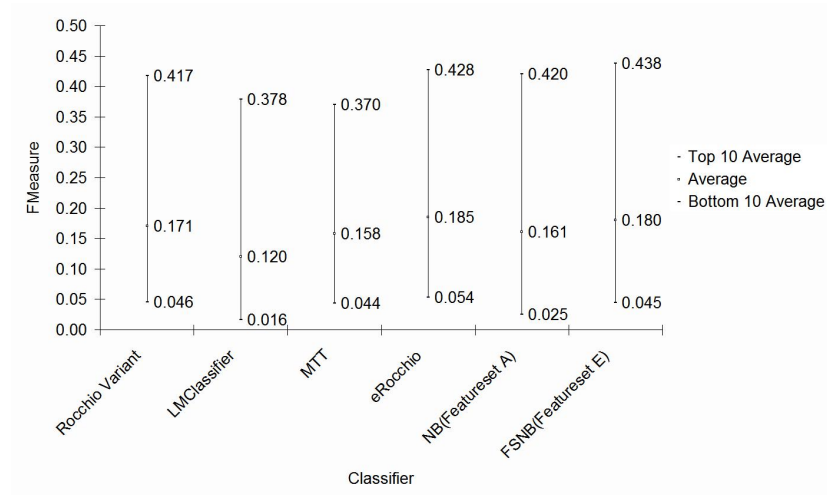
Figure 9.8: Bottom-10, top-10, and complete average FMeasure for the Digg dataset.

with feature selection and the expanded featureset begins to perform better than the Rocchio variant halfway through the experiment. The figure also suggests that iScore in its original configuration does not perform as well as the Rocchio variant. However, Figure 9.10, which shows the classifiers' FMeasure performance over the 5,000 most recent documents at various time periods, paints a difficult picture. The figure indicates that iScore with feature selection and the expanded featureset performs significantly better than all the other classifiers with the exception of the first time period. The figure also shows that there is less of a drop-off in performance seen in the latter time periods for iScore than the other classifiers, indicating better stability against noise and gaps in data collection.
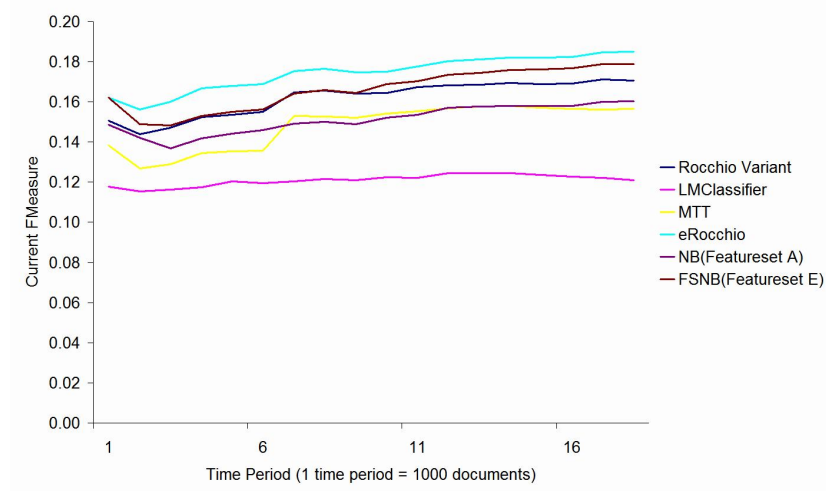
Figure 9.9: Current cumulative FMeasure at specific periods for the Digg dataset.

9.4 Summary

In summary, iScore with online feature selection operating on the original iScore featureset, MTT, and eRocchio works generally well for all datasets. The Yahoo! News and tagger datasets clearly supports this hypothesis in all tests. The Digg dataset also support this hypothesis when one looks at the FMeasure performances of the classifiers over the the 5,000 most recent documents at various time periods. Given that all three of the datasets show the value of the expanded featureset and all three datasets support the value of online feature selection, it can be concluded that online feature selection and the expanded featureset in the iScore framework can generally recommend better articles than traditional information retrieval techniques.
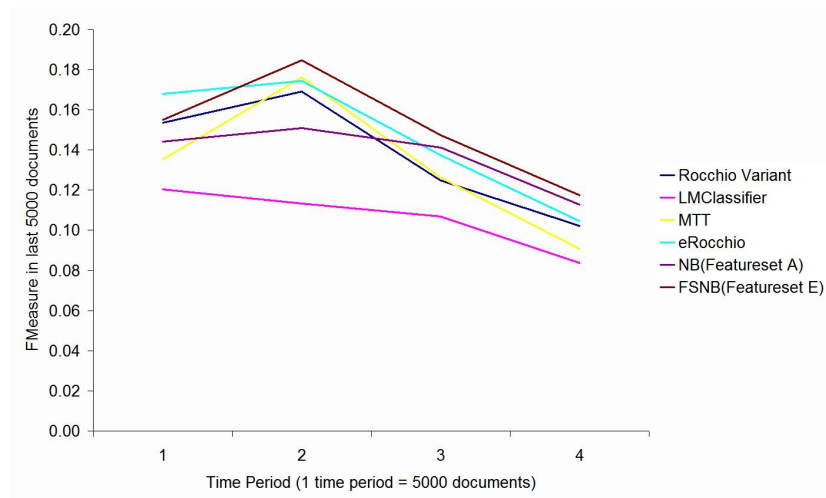
Figure 9.10: FMeasure for the 5,000 most recent documents for the Digg dataset. iScore with feature selection and the expanded featureset performs significantly better than all the other classifiers with the exception of the first time period.

CHAPTER 10

Future Work

In this chapter, several possible roads of research are discussed that will either extend the news recommendation framework discussed in this dissertation or may improve the recommendation performance of iScore.

10.1 Other user models

Instead of filtering articles as they are published so that only interesting articles are presented to the user, articles could be ranked in order of interestingness. For example, a list of articles can be presented to the user after he enters a search request either by a query string [Goo08] or by a faceted-search request [KZL08]. However, this "interestingness" problem is a different problem from the "interestingness" filtering problem addressed by this dissertation so not all features will be applicable, the semantics of some features will change, and other features will be necessary. For example, recent articles are more likely to be interesting in the search-model since articles from various time periods may be returned for the same request; whereas, articles in the filter-model are always the most recent articles since they are analyzed as they are published.

A faceted-search interface can also be extended to include the various features of interestingness discussed as facets. For example, if a user is looking for only "subjective" articles, a user can indicate his interest by selecting the "subjective"
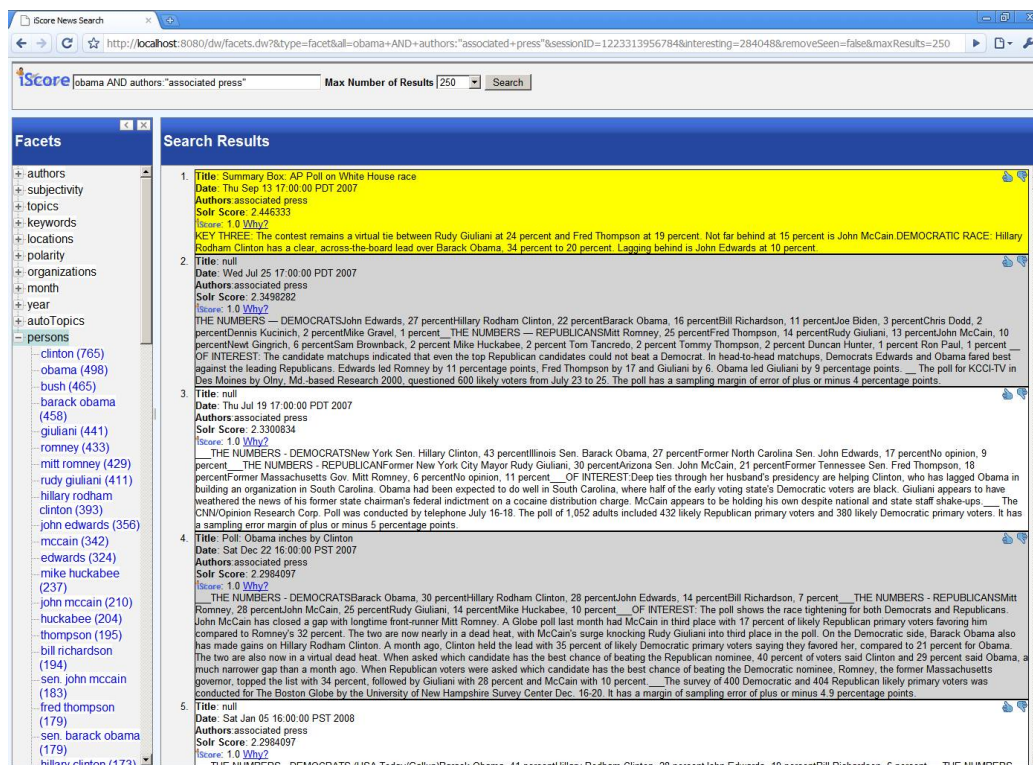
Figure 10.1: Solr configured for the Yahoo! News dataset, using subjectivity, polarity, named entities and topic clusters as facets. iScore has been integrated into this Solr application that allows for the reordering of search results.

facet of the search interface. A good starting point would be to precompute user-independent features, such as subjectivity. These features may be indexed to enable fast lookups. Then when a user issues a query, a simple intersection with the topic-specific facets and the "interestingness"-driven facets would generate the search results list.

Another extension of iScore is to apply the lessons learned in iScore into developing an interactive news search, where queries are issued either by a query string or by facet selections. Results are returned in order of interestingness. A user may then tag articles with his opinion regarding the interestingness of the

article and the ranked-list is dynamically updated. This will necessitate the use of very fast feature extractors and a fast updateable classifier as the response time of the query is now more crucial since the user will want results as soon as he issues his query. Additionally, it will not be possible to reprocess all articles given a user tagging, so articles that are relevant to his query will be the only ones to reprocessed. Dynamic pruning and addition of the document-space will be necessary since a user's tagging may influence the size and coverage of the document-space. Identifying this document-space will be a significant challenge as it will pose several challenges:

1. Too large of a document-space will slow down response time.

2. Identifying articles with user-dependent features (e.g., topic-relevancy) that are not likely to be significantly influenced by a user's new tagging may be necessary so that they are not included in the document-space for reprocessing.

3. The cost of calculating the document-space should be relatively low so that user-response time is kept low.

A demo, using the Solr search server [Sol08], using some of these features is shown in Figure **??** including subjectivity and polarity ratings, topic clusters generated by Cluto [Kar03] and named entities generated by the Stanford NER [FGM05] for the Yahoo! News dataset. A user may indicate his interest in an article by clicking the "thumbs-up" or "thumbs-down" signs. If a user clicks on the "thumbs-up" sign, then the article associated with the tagging will be labeled as interesting and all the article preceding it (without a tagging already) will labeled as uninteresting. The reasoning behind this inferred user-model is that a user will scroll down the search results until he finds an article he is interested

in, making all the preceding articles, that have not already been tagged, uninteresting. If a user clicks on the "thumbs-down" sign, then the article associated with the tagging will be labeled as uninteresting and all the articles preceding it (without a tagging already) will be labeled as interesting. The reasoning behind this user model is similar to that of the reasoning behind the "thumbs-up" sign. By inferring user taggings, the classifiers and the feature extractors are trained quickly without much user-interaction. This demo utilizes a naïve Bayes classifier as its overall classifier, using the following feature extractors:

1. Interestingness counts for cluster membership

2. Language models operating on the title and the body of the article

3. eRocchio

4. Freshness

5. Interestingness counts for keywords (i.e., the top-5 terms with the highest TF-IDF values)

6. Interestingness counts for named entities (i.e., locations, organizations, and persons)

7. Polarity

8. Subjectivity

9. Source Reputation

10. Interestingness counts for RSS Feed membership

An alternative method for returning search results in order of interestingness is to pre-cluster the articles in the document collection on features that are not

user-dependent, such as news sources, term-frequencies, sentiments, uniqueness, and publication times. When a user issues a query for articles, a search results list returns, giving him the articles relevant to his search. Each of the articles returned to him will have a pre-computed cluster of articles associated with it. When a user indicates his interest in an article, the search results list can be reordered such that the articles belonging to the interesting article are ranked higher than they were before. As more articles are marked as interesting, articles from clusters encompassing the majority of the interesting articles are ranked higher. This avenue of research will pose several issues:

1. Which clustering algorithm will be appropriate? Will a hierarchical clustering algorithm allow the user to narrow in on the collection of articles he is interested in?

2. How and by how much should the rank of articles belonging to recommending clusters be increased?

3. How will the recommending clusters be determined? Will selecting all clusters containing all interesting articles result in too large of a document-space? Will selecting clusters containing only the majority of interesting articles result in too small of a document-space?

4. If a hierachical clustering approach is used, it is likely that an interesting article is contained by multiple clusters of different sizes at multiple levels which encompass smaller clusters that are lower in the hierarchy. Which cluster level and/or size is appropriate?

5. How will the clusters be indexed so that they are easily found when the results list are reordered after a user indicates his interest for an article?

6. Will multiple clusterings on different featuresets help?

10.2 Incremental conditional classifiers

In this dissertation, it is assumed that "interestingness" features are conditionally independent so that an incremental and fast classifier, such as naïve Bayes, can be used. Although, performance of naïve Bayes has shown to be good, performance may be improved by employing another classifier that does not assume that features are conditionally independent. Such classifiers already exist, such as decision trees, but they are not incrementally updateable. A neural network [WF05] is a possible candidate as the weights of the input features can change over time and a multi-layered network can address the conditional dependence issue. However, the structure of the network is neither trivial to construct nor updateable. A Bayesian network classifier [WF05] is a generalized version of a naïve Bayes classifier, but does not assume conditional independence among all the features. Like that of a neural network, a structure of a Bayesian network is neither trivial to construct nor updateable.

Current classifiers are not sufficient to fully address the classification problem in this dissertation. A solution to this classification problem will have to address the following issues:

1. Be easily updateable, not requiring complete retraining of the classifier.

2. Does not require maintaining a complete history of data items for updating.

3. Be able to identify conditional dependence among features dynamically.

10.3 Incorporating semantic information

In general, topic relevancy features are shown to be one of the most important features for identifying interesting articles. The current approaches used assume a bag of words approach, which fails to include semantic and background information that may be useful to determine the actual meaning of an article. It has been shown that by incorporating semantic information in document classification, that classification performance can improve [GR08].

One method for incorporating semantic information is explicit semantic analysis (ESA) [GM07], which extracts background information from external sources, such as Wikipedia [Wik08]. A semantic interpreter maps fragments of natural language text into a weighted sequence of concepts (as defined by Wikipedia) ordered by their relevance to the input. Consequently, a text fragment is represented as a weighted vector of concepts. A classifier can be trained on the content of Wikipedia to determine the weights of concepts for a text fragment. Therefore, the meaning of a text fragment is interpreted in terms of its relatedness with Wikipedia concepts. These concepts may be concatenated with a TF-IDF document vector along with topics extracted from LSA (see Appendix B) to create a much larger and rich feature vector so that algorithms, such as eRocchio or MTT may be performed. This much larger and rich feature vector will incorporate external information not present in the article that would help with the understanding of the article. This is a very similar method as discussed in [GR08]. An alternative method for including this semantic information in the "interestingness" classification problem may be to push these weighted concepts up to the overall classifier, each as its own individual feature.

10.4 Analysis of changes in sentiment

Sentiment analysis has already been included in the iScore framework by including as features, the outputs from the language model classifiers that classify sentences' objectivity and polarity. Although, users may have a particular preference for an article's objectivity or polarity and it is useful for judging the "interestingness" of an article for a user, the change in objectivity and/or polarity from the expected behavior of a source or topic may be additionally useful. For example, if a news source is consistently known to provide objective material in its news article and if subjective news articles begin appearing, this may indicate the birth of an interesting news event. To illustrate, if a news writer, who is known to publish objective news reports on a topic, such as a political campaign, suddenly begins publishing subjective news reports, possibly condemning a campaign, then some type of interesting news event must have sparked that change in material, such as a political scandal or a political gaffe. Additionally, changes in the polarity of a news topic may indicate event changes that would result in interesting articles being published. For example, if the opinion of articles, from a specific news agency, on the 1996 Atlanta Olympics has generally been positive and then suddenly turns negative, then some type of interesting news event must have happened, such as the bombing incident that occurred during those Olympic games. In addition to news sources, changes in sentiment may also be tracked for the opinions of organizations or persons on other organizations or persons or places. The sentiment surrounding named entities can be tracked over time for this feature.

10.5　Future trends

With the advent of blogs that specialize in niche news markets, readers can expect to see an explosive growth on the availability of information where only a small fraction may be of interest to them. In contrast to traditional news sources, such as CNN, blogs focus on specific topics that may be of interest to only a handful of users as opposed to the general public. This phenomenon is often referred to as the long tail market phenomenon [And07]. Instead of building news filters that cater to the mass public, future research will focus more on personalized news recommendation. Personalization research is also present in other media, as evident in the Netflix Prize competition [Net07] and the related Knowledge Discovery and Data Mining Cup 2007 competition [BEL07], in which teams compete to improve the accuracy of movie recommendations.

Traditional corpora, such as the ones used in TREC, are ill equipped to address the problems in personalized news recommendation. Current corpora address the traditional problems of topic relevancy and do not address the problem of interestingness. Furthermore, such corpora are not user-focused. At best, such corpora label articles that a general audience would find to be interesting as opposed to a specific user. Further research in personalized news recommendation will need to be evaluated on a large test data collection that has been collected using many individual users. Personalized news portal such as those provided by Yahoo! and Google often track which articles are read and rated by specific users. It is hypothesized that in time, as user personalization becomes more prolific, test collections from organizations, such as Yahoo! and Google, will become more publicly available that it will allow easier comparison among recommendation algorithms.

CHAPTER 11

Conclusion

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

To address news recommendation in a limited user environment, where methods such as collaborative filtering would perform poorly, a news recommendation framework, called iScore, is introduced to analyze the reasons why an article is interesting. The iScore framework consist of several advances. Naïve Bayes working over several key features is identified to be a generally good classifier for this news recommendation problem. A new feature to address the multiple topics that a user may be interested in is also introduced. However, one of the drawbacks with the method of tracking multiple topics of interest to a user is that parameters often must be decided before the algorithm becomes active. To address this problem, online parameter selection in Rocchio, simplifying the technique used in multiple topic tracking, is introduced. It can attain similar or even better recommendation results as the best baseline classifiers that require extensive parameter tuning. Although many different features are available to help with classification, not all features are equally useful for users. Consequently, online feature selection in naïve Bayes that would be appropriate to the problem of on-

line news recommendation results is studied. By incorporating these features and online feature selection, iScore can generally give better recommendation results than standard information retrieval techniques in the three datasets that iScore has been evaluated with: the Yahoo News!, tagger, and Digg collections.

Although, iScore has been evaluated on several different datasets, the datasets used are difficult to collect and not freely open for distribution. There is still a need for a more agreed upon and open rigorous evaluation framework so that different recommendation systems can be evaluated against one another. This will aid the research community to make more advances in this area of work.

In summary, the following contributions were made in this dissertation towards an online news recommendation system:

1. Identification of high-level features interestingness, such as topic relevancy, uniqueness, source reputation, writing style, freshness, subjectivity, and polarity.

2. Implementations of the high-level features.

3. Ensemble technique using naïve Bayes classifier to incorporate user feedback.

4. Multiple topic tracking to address the multiple topics of interest that a user may find interesting.

5. Online parameter selection for the Rocchio, to aid in user-tailoring of algorithms.

6. Online feature selection naïve Bayes, to address the varying reasons for interestingness among users.

7. High quality recommendation results.

It is the combination of all these techniques that yield as much as 50.7% better recommendation results than traditional information retrieval techniques. More specifically, for the Yahoo! News dataset, there is a 24% improvement over the baseline classifier. For the tagger dataset, there is a 50.7% improvement. And for the Digg dataset, there is a 5% improvement.

APPENDIX A

Implementation

iScore is implemented with an assortment of tools in Java. LingPipe [Ali06] is used for building language models and related classifiers. OpenNLP [Ope06] is used for sentence detection and noun phrase extraction. Clustering is accomplished with Cluto [Kar03]. The Stanford Named Entity Recognizer [FGM05] is used for named entity extraction. Other classifiers are from Weka [WF05]. The majority of the iScore experimental setup is developed with the Apache Unstructured Information Management Architecture (UIMA) framework [UIM08], which allows for the plug-and-play ability of various text analysis engines and components together in a single pipeline.

A.1 Producers

The NewsItemProducerFromFile2 producer reads and provides articles and its associated metadata to the rest of the UIMA pipeline. The producer reads from local files to generate UIMA objects for each article that are read and consumed by text analysis engines and consumers, respectively. These objects are defined in Table A.1.

All UIMA objects, except ParsedContent, are constructed by iterating over files associated with its type that contain all information for all articles in an order determined by the file document_order. The file document_order contains

Object	Description
NewsItem	Title and publication date
ParsedContent	Body
Author	Author names or Hostname of article
Cluster	Parent cluster ID and similarity to cluster
FeatureScore	Feature ID and value
ClassifierScore	Classifier ID and value
NamedEntity	Named entities contained in body and title
NounPhrase	Noun phrases contained in body and title
Tagging	Interestingness tagging by a user

Table A.1: Objects generated by the NewsItemProducerFromFile2 producer.

document identifiers sorted in the order that the documents will be processed. A ParsedContent object is created by reading a single file associated with the document identifier.

Other producers have been developed for the collection of articles, such as from Digg, Yahoo! News, and tagger datasets.

A.2   Text analysis engines

There are three kinds of text analysis engines (TAE) in the iScore architecture: FeatureScorers, Classifiers, and Thresholders. Each text analysis engine, during its initialization phase, generates its own unique identifier by querying the database. The FeatureScorers take as input various raw data objects, such as NewsItem, ParsedContent, Author, Cluster, NamedEntity, NounPhrase, and Tagging objects, and generate feature values for a document and produce a Fea-

tureScore object. The Classifiers take in FeatureScore, NewsItem, and Tagging objects as input and generate a ClassifierScore object which contains the probability that the article is interesting. The Thresholder takes in ClassifierScore, NewsItem, and Tagging objects as input and generate an Interestingness object that contains the binary decision regarding the interestingness of the article.

A.3   Consumers

There are three kinds of consumers in the iScore architecture. The FeatureScore and ClassifierScore consumers record their respective objects' data in a file and then import them later in a MySQL database for permanent storage. The Statistics consumer consumes Interestingness objects and Tagging objects and generates and stores statistics about each run, such as FMeasure, precision, and recall in the MySQL database. The FeatureScores and ClassifierScores stored in the MySQL database are exported to files so that they may be entered into a Classifier or Thresholder pipeline by the NewsItemProducerFromFile2 producer.

A.4   Pipelines

The overall iScore architecture described in Figure 4.1 is broken down into three kinds of pipelines in the iScore experimental framework to aid in debugging and experimentation, as shown in Figure A.1. For each user and feature (or user and classifier, or user and thresholder) pair in the experimental setup, there is a pipeline generated. Because the runs are embarrassingly parallel, pipelines are run on a multi-node cluster, to hasten the acquisition of experimental results.

The first type of pipeline is the FeatureScorer pipeline, which contains the NewsItemFromProducerFile2 producer, a FeatureScorer, and FeatureScore con-

sumer. The second type of pipeline is the ClassifierScorer pipeline, which contains the NewsItemFromProducerFile2 producer, a Classifier, and ClassifierScore consumer. The final type of pipeline is the Thresholder pipeline, which contains the NewsItemFromProducerFile2 producer, a Thresholder, and Statistics consumer. It is expected that several FeatureScore pipelines are run first to generate feature scores, followed by the Classifier pipeline to generate the overall classifier scores, and followed by the Thresholder pipeline to generate statistics about the final run.

The source code for all the TAEs, consumers, and producers are located at `http://sourceforge.net/projects/iscore/`. The pipelines were executed on over 50 individual Linux machines and then later on a 80-node Linux cluster with dual-processors for each node, and a 20-node cluster with quad-processors for each node. For each dataset, approximately one week of computation time is necessary to run all the experiments detailed in this dissertation.

A.5 Data collection

The Yahoo! News and the Digg datasets are collected with a UIMA pipeline, running every three hours on a Linux machine at Lawrence Livermore National Laboratory (LLNL). The tagger dataset is collected by asking volunteers to use a Firefox plug-in or a GreaseMonkeyScript for the Google RSS Reader, both found at `http://goliath.cs.ucla.edu/tagger`, which records webpages they have read and not read on a server at UCLA. The webpages pages are downloaded nightly by the server. A web tool, written in PHP, located at `http://goliath.cs.ucla.edu/clean/cleanDocument.php`, is used to discard non-news webpages.
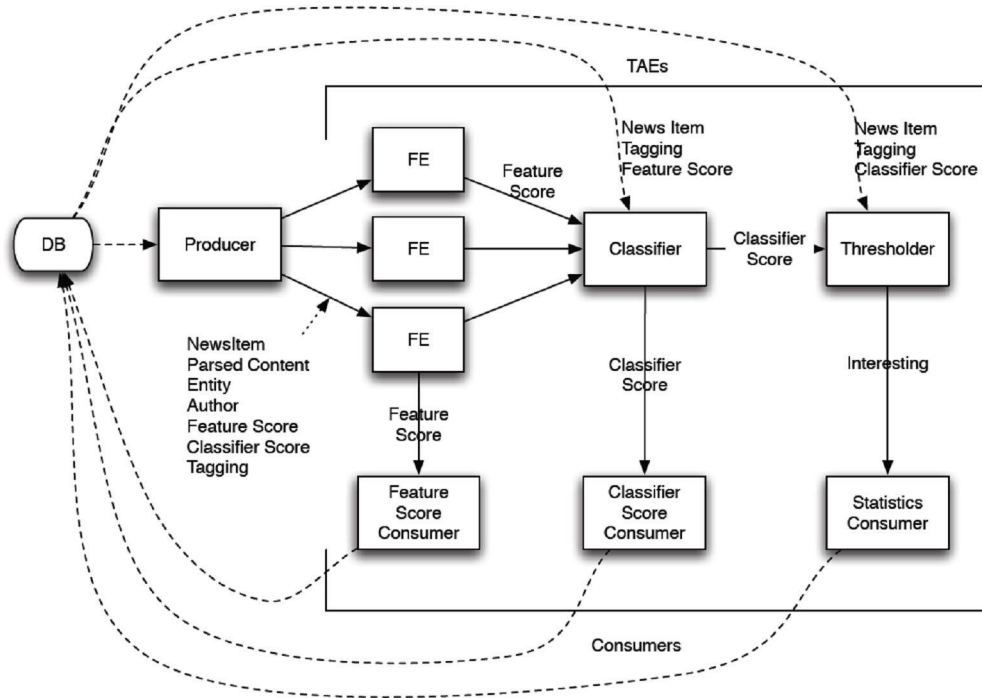
Figure A.1: The overall iScore architecture partitioned into three separate pipelines for experimentation. Output from each stage is stored in the database. The database contents are dumped into a text file before being fed to a producer.

APPENDIX B

Other Roads of Research Considered

In this appendix, roads of research are discussed that did not improve iScore's recommendation performance in the experiments. However, by documenting what was attempted, future work can avoid repeating what did not work or will attempt to work on similar ideas but with different angles.

B.1 Multi-role users

The work in MTT shows that users are interested in a multitude of news topics and that not one single model can accurately capture all the interests of the user. This may also be applicable to other features. Depending on the role that a user is serving while reading the news, some features may be interpreted differently depending on the role the user is currently serving. For example, a user interested in "terrorist activity" would have a higher preference for negative polarity articles; whereas, when the same user may also be interested in "UCLA basketball" would have a higher preference positive polarity articles.

Initial experimentation is done with the small Yahoo! News dataset and only on the "Top Stories Politics" feed but did not yield fruitful results. Two techniques are experimented with based on this concept, hoping that it would improve recommendation results: (1) building topic-specific classifiers that are only trained on documents belonging to its parent-topic, and (2) building topic-

specific classifiers that are trained on all documents but with different weights proportional to a document's relationship to the classifier's parent-topic. With technique 1, the data used for training the classifiers were much too sparse to be effective. With technique 2, the computational cost for training the number of classifiers necessary to represent each topic are too costly.

B.2 Identifying interesting relationships and entities

The use of entity networks or ontologies representing a user's interests may be useful for differentiating interesting articles from uninteresting ones. [Ang05] presents a survey of various techniques necessary for ontology acquisition, including named entity extraction, the acquisition of concepts, and learning taxonomies. [Sin04] studies how binary relationships are expressed within single sentences. In each sentence, pairs of target entity types are considered as candidates and are classified with a support vector machines with a variety of features. A preliminary study has been done to evaluate the applicability of using PageRank [PBM98] to identify important entities in news, such as persons or locations. Using this information, the most interesting articles can be identified that discuss these entities. This task entails the following three subtasks:

1. Construct a meaningful directed graph, where two connected entities are connected by a directed edge such that the source of the edge is transferring some of its "importance" to the destination of the edge. The relationship between the entities should be learned from articles or from an external knowledge-base.

2. Evaluate PageRank on the network to determine the importance of the entities in the network.

160

3. Using the entities' PageRank scores, evaluate any new article's "interest-ingness."

As a preliminary study, the relationship of the PageRank scores of entities to the interestingness of their containing articles on the small Yahoo! News dataset for the "Top Stories Politics" feed is studied. A graph of entities is maintained as documents are processed in chronological order of their publication date. Using Sundance/AutoSlog [RP04], a shallow parser and pattern extractor NLP tool, the subjects and direct objects of sentences were extracted from each article. The subjects and direct objects extracted are added to the graph and a relationship is also added, connecting the subject to the direct object. Intuitively, the subject is passing some of its "importance" to its direct object because it is "doing something" to its direct object. For example, in the sentence, "George Bush lunched with the Boy Scouts of America," George Bush is passing some of his importance to the Boy Scouts of America because the Boy Scouts are important enough for Bush to have lunch with them. However, it can be argued that the inverse is also true. But for this preliminary evaluation of PageRank's applicability and for simplicity, the subject to direct object relationships is only considered. Additionally, for each entity, a count is maintained of how many articles contain the entity and how many interesting articles contain the entity as well. Using these two counts, a TF-IDF statistic is maintained for each entity e:

$$TF - IDF_{Interesting}(e) = \frac{|\text{Int Articles containing } e|}{|\text{Int Articles Seen So Far}|} \log(\frac{|\text{Articles Seen So Far}|}{|\text{Articles containing } e|})$$
(B.1)

Processing articles in chronological order of their publication date, the PageRank scores of all entities in the current network is computed every 7 days. However, instead of the traditional PageRank scores detailed in [PBM98], PageRank with

161

priors [WS03] is used to give more weight to entities that that are known to be more important to the user. In [WS03], PageRank is formulated as the following, where $d_{in}(v)$ is the set of entities that point to an entity v and $p(v)$ is the prior bias of v:

$$PageRank(v) = (1 - \beta) * \left(\sum_{u \in d_{in}(v)} p(v|u) PageRank(u) \right) + \beta * p(v) \qquad \text{(B.2)}$$

In [WS03], β=0.3 and $p(v) = \frac{1}{|R|}$ for $v \in R$, where R is the set of known important entities, $p(v)$=0 otherwise. The same configuration is used in the experimental evalutions. Whenever the periodic PageRank scores $PageRank_t$ for time period t are computed, the set of known important entities are the current top 100 entities with the highest TF-IDF scores of the last 7 days. After all the documents are processed, the PageRank scores $PageRank_c$ of all the entities are computed on the complete final network using the top 100 entities with the highest TF-IDF scores overall. For each document, the features detailed in Tables B.1 and B.2 are extracted, where E is the set of all entities contained within a document, t is the last time period for which PageRank was computed and $IDF(e) = \log(\frac{|\text{Articles Seen So Far}|}{|\text{Articles containing} e|})$.

To evaluate how well PageRank could be used to infer the "interestingness" of articles, the features extracted using Weka's Knowledge Flow tool with the flow shown in Figure B.1 are studied. A 10-fold cross-validation is run on the dataset for the interesting articles defined by the "Top Stories Politics" feed and used AdaBoosting [FS96] on a C4.5 decision tree generated by the J48 algorithm.

The precision-recall curve for the classifier using the extracted features to predict the interestingness of news articles is shown in Figure B.2. The graph shows that it is possible to attain 50% precision, with 10% recall. Using two feature selection criteria: information gain and chi-squared rankings, The usefulness of the

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

Table B.1: PageRank-based features, Part 1
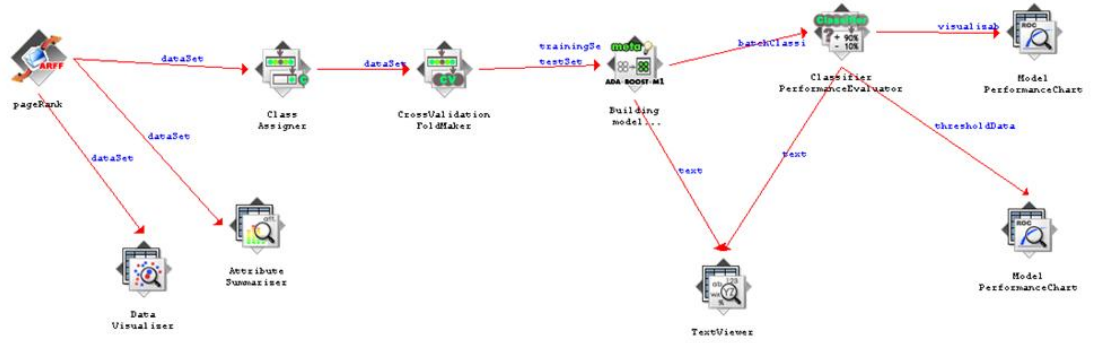
Figure B.1: Knowledge Flow in Weka.

163

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

Table B.2: PageRank-based features, Part 2
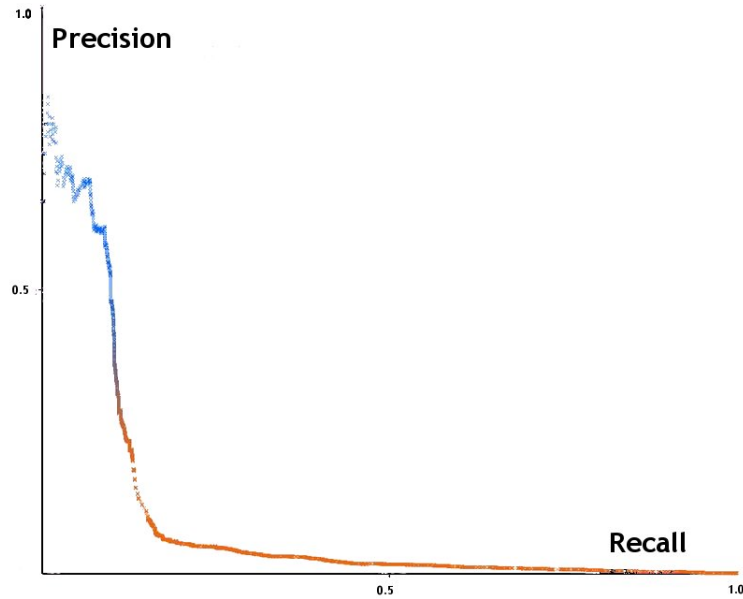
Figure B.2: Precision-Recall curve for PageRank featured classifier.

PageRank features is determined. For the information gain rankings, minDelta, maxDelta, sumDelta, maxIDF, and maxLogIDF were the top 5 most useful features. For the chi-squared rankings, minDelta, maxDelta, sumDelta, avgDelta, and stddevDelta were the top 5 most discriminating features. These two rankings indicate that the changes in the importance of entities in news are more useful than static PageRank scores for determining which articles are interesting.

Although, PageRank features shows some promise for helping determine the "interestingness" of articles in the initial evaluation, when incorporating these PageRank features into iScore by experimenting with the "Top Stories Politics" feed from the small Yahoo! News dataset, several problems are encountered:

1. Despite the promising performance when used in a J48 decision tree, the PageRank-Change feature when added to the iScore framework as an additional feature, did not yield any improvement on top of the recommendation

performance of iScore with the original feature set.

2. The entities extracted are not easily co-referenced. As a result, the entity graph is too large to manipulate in an online setting. Methods for pruning the graph are employed to remove entities that only appeared once in a time period, but the resulting graph is very sparse.

Advance co-referencing techniques that can be accomplished in an online setting will need to be employed for this area of research to become more fruitful.

B.3   Latent semantic analysis

The current approaches to topic relevancy used assumes a simple bag of words approach, which fails to include semantic and background information that may be useful to determine the actual meaning of an article. One method for incorporating semantic information is latent semantic analysis (LSA) [Ste06]. One technique for LSA experimented with is a generative model that uses a Dirichlet distribution by Steyvers. In this model, a document is treated as a mixture of topics. For a particular document, let $P(z)$ be the distribution of topics z and $P(w|z)$ be the distribution of words w given topic z. The generative model used specifies the following distribution over words within a document:

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j)P(z_i = j) \qquad \text{(B.3)}$$

where T is the number of topics, w_i is the i-th word, $P(z_i = j)$ is the probability that the topic of the i-th word sampled is the j-th topic, and $P(w_i|z_i = j)$ is the probability of word w_i under topic j. Gibbs sampling with a Dirichlet prior is used to estimate the distributions $P(z)$ and $P(w|z = j)$.

In the experimental evaluation, the probability distributions are estimated by

processing the small Yahoo! News dataset using the Gibbs sampling approach for 300 topics. Then for each new article in the the large Yahoo! News dataset, a vector of topic probabilities are estimated by sampling the probability distributions generated earlier. The vector of topics are then used in a Rocchio-like scheme, where $\vec{p}_T$ is the sum of all topic vectors of interesting articles and $\vec{d}_T$ is the topic vector of the current document:

$$f_{LSA}(d) = \cos(\vec{p}_T, \vec{d}_T) \tag{B.4}$$

Unfortunately, the FMeasure performance of this feature alone is less than that of Rocchio operating on the bag of words alone. This lower performance may be due to several issues. One possible reason for the lower performance is that the number of topics generated from the pre-processing step is too small. Also, the number of samples used to estimate the new topic distributions of each new article is not large enough. A small number of samples is used so that the experiments could be accomplished in a timely manner. Another possible issue is that in reality, topics do not stay static. New topics may be born or may continually evolve. Unfortunately, the method used does not allow for new topics to be added or for existing topics to change efficiently without reprocessing all articles seen.

B.4 Cluster interestingness

Another topic relevancy measure experimented with is based on measuring the interestingness of clusters. Clusters generated with Algorithm 1 and bootstrapped with clusters generated by Cluto [Kar03] over the small Yahoo! News articles dataset. Cluster interestingness for an article d is defined as the weighted sum of the interestingness of clusters that d has a cosine similarity greater than 0.4

with:

$$f_{ClusterInterestingness}(d) = \frac{\sum_{c \in Clusters(d)} \cos(\vec{c}, \vec{d}) * Interestingness(c)}{\sum_{c \in Clusters(d)} \cos(\vec{c}, \vec{d})} \qquad \text{(B.5)}$$

where the interestingness of a cluster is the proportion of articles belonging to the cluster that have been marked as interesting:

$$Interestingness(c) = \frac{|\text{Int articles in } c|}{|\text{Articles in } c|} \qquad \text{(B.6)}$$

However, when the Cluster Interestingness feature is added to the iScore framework, FMeasure performance drops. This is most likely due to the fact that Cluster Interestingness is measured very similarly and more accurately by other features such as MTT. Consequently, the addition of Cluster Interestingness to iScore is simply noise to the overall classifier.

B.5   Cluster popularity

In addition to measuring the interestingness of clusters, the popularity of clusters wasis also experimented with. Intuitively, topics written often by authors are more likely to be interesting than topics that are infrequently written about. The popularity of a cluster is the proportion of articles published since the creation of the cluster that belong to the cluster. Thus, the Cluster Popularity feature value for an article d is the popularity of the cluster that d has the maximum cosine similarity to:

$$f_{ClusterPopularity}(d) = \frac{|\text{Articles belonging to } c_{max}|}{|\text{Articles belonging published since } birth(c_{max})|} \qquad \text{(B.7)}$$

However, when the Cluster Popularity feature is added to the iScore framework, FMeasure performance drops. This is most likely due to the fact that Cluster Popularity, in general, does not identify additional interesting articles

that have not already been identified by the existing features in iScore, but instead adds additional noise to the overall classifier.

B.6   Entity interestingness

Like, the Phrase Interestingness feature discussed in Chapter 7, the Entity Interestingness feature aims at addressing the shortcoming of the bag of words approach. Named entities are extracted using the Stanford Named Entity Recognizer (NER) [FGM05]. Named entities are looked at instead of general noun phrases because it is hypothesized that named entities extracted by the Stanford NER are generally more accurate than a noun phrase extractor and are more representative of the most important noun phrases in a body of text. The feature measures the average probability of the interestingness of named entities as:

$$f_{EntityInterestingness}(d) = \frac{\sum_{p \in entities(d)} \frac{|\text{ times } p \text{ occurs in Int articles}|}{|\text{times } p \text{ occurs in articles}|}}{|phrases(d)|} \qquad \text{(B.8)}$$

However, when the Entity Interestingness feature is added to the iScore framework, FMeasure performance drops. This is most likely due to the fact that Entity Interestingness, in general, is addressed by Phrase Interestingness since named entities are a type of noun phrase, which are already examined by Phrase Interestingness. Due to this result, it can be concluded that the original hypothesis about named entities being the best noun phrases to represent a body of text is false. There are other noun phrases that are not named entities that may be required to fully understand a body of text. For example, in the following sentence, there are no named entities:

The brown dog jumped over the fence.

The noun phrases "the brown dog" and "the fence" are necessary to understand

the meaning of the sentence, but are not named entities.

B.7   Significant n-grams

To address the poor performance of iScore over the initial set of articles due to the learning that must be accomplished before accurate results are given, information regarding universally interesting articles is bootstrapped. More specifically, the n-grams that appear more often in the topic-independent and interest-driven feeds from the small Yahoo! News collection than in other feeds are looked at. Intuitively, there should be some articles that most users would find universally interesting, such as the first articles about the 9/11 attacks. Significant n-grams from topic-independent feeds are determined in a similar fashion to how language models are used for identifying new n-grams for anomaly detection as described in Chapter 4. A background model is built on all articles in the small Yahoo! News collection. The foreground model is built on all articles from the topic-independent feeds from the small Yahoo! News collection. The foreground and background models are compared with one another and the significant n-grams of sizes one to five (where the grams are words) are extracted.

Given this list of significant n-grams and their significance (based on the z-score as discussed in Chapter 4), the overall user-independent interest of an article is measured as:

$$f_{SignificantNgrams}(d) = \sum_{g \in sigNgrams(d)} significance(g) \qquad \text{(B.9)}$$

where $signNgrams(d)$ is the set of significant n-grams generated by the comparision of the foreground/background language models discussed earlier contained within the article d. The significance of an n-gram g is denoted by $signficance(d)$.

Like the Cluster Popularity feature, the Significant N-grams feature did not

improve the FMeasure performance of iScore when added to the framework because it did not identify any articles that have not already been identified by existing features; thereby, adding noise to the overall classifier.

B.8 Tracking new n-grams

Another n-gram-based feature experimented with looks at tracking the new and hot n-grams, which is another form of uniqueness. As articles are analyzed for classification, a foreground model built upon the previous week's articles is compared to the background model built upon the news published in the week prior to the publication date of the articles in the foreground model. The top-k n-grams (where k is 100 n-grams and the size of the n-grams that are considered ranged from one to three in the evaluations) that appear more often in the foreground model are identified. Intuitively, articles that contain these new n-grams are more likely to be interesting because these n-grams are related to hot and new topics. Therefore, the Tracking New N-grams feature is defined similarly as the Significant N-Grams feature as defined by Equation B.9, except with a different $sigNgrams$ set. In addition to looking at all articles when generating the $sigNgrams$ set, another variation of this feature looks at interesting articles only, instead of all articles.

However, both variants of the Tracking New N-grams feature fail to improve iScore's recommendation performance. The user-independent variant shows poor correlation to interestingness. The user-dependent variant did not identify any articles that have not already been identified by existing features, particular the topic-relevancy feature; thereby, adding noise to the overall classifier.
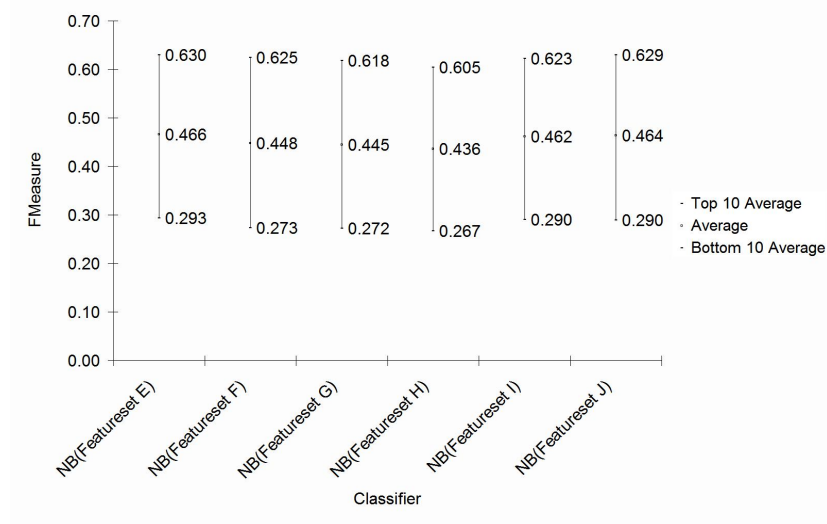
Figure B.3: Bottom-10, top-10, and complete average FMeasure for the best featureset and other featuresets for iScore on the Yahoo! News dataset.

B.9 Experimental results

Cluster interestingness, cluster popularity, entity interestingness, significant n-grams, and tracking new n-grams are experimented with the Yahoo! News, tagger, and Digg datasets. The results are shown in Figures B.3, B.4, and B.5. Results of other roads of research considered are not available because they were deleted after their initial evaluation. Table C.2 in Appendix C describes the featuresets in the figures.
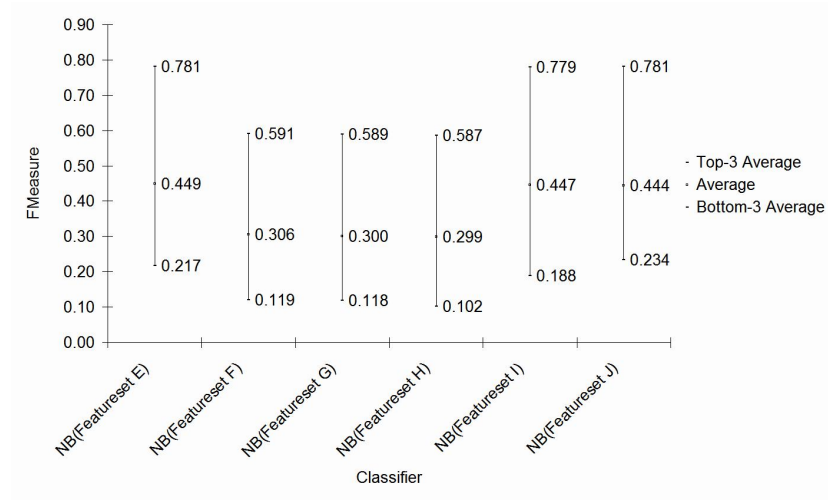
Figure B.4: Bottom-3, top-3, and complete average FMeasure for the best featureset and other featuresets for iScore on the tagger dataset.
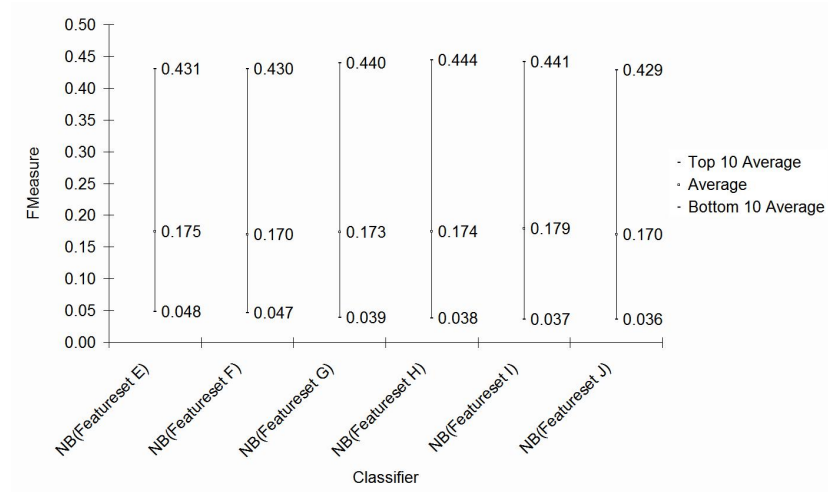
Figure B.5: Bottom-10, top-10, and complete average FMeasure for the best featureset and other featuresets for iScore on the Digg dataset.

APPENDIX C

Reference Tables

ID	Description
A	AuthorStyle, BinaryLMClassifier, Rocchio Variant, Rocchio, DirectSubjectiveLMClassifier, Freshness, IncrementalClusterAnomalyDetection, LMAnomalyDetection, LMClassifier, NaïveBayesLMClassifier, NewNGrams, ObjectiveSpeechEventLMClassifier, PolarityLMClassifier, SourceReputation, SubjectivityLMClassifier
B	Featureset A, MTT
C	Featureset A, eRocchio
D	Featureset B, eRocchio
E	Featureset D, PhraseInterestingness, TitleLanguageModel, BodyLanguageModel, TitleAnomalyLanguageModel, BodyAnomalyLanguageModel, TopicDrivenFreshness, ClusterMovement, SlidingAnomalyDetection

Table C.1: All featureset identifiers and descriptions, Part 1.

ID	Description
F	Featureset E, TrackNewGrams, SignificantNGrams
G	Featureset F, EntityInterestingness
H	Featureset G, ClusterInterestingness
I	Featureset E, EntityInterestingness
J	Featureset E, ClusterPopularity

Table C.2: All featureset identifiers and descriptions, Part 2.

Usernames		
aaaz	Aidenag	antdude
BoneyB	bonlebon	Brajeshwar
cambrown99	capn_caveman	chad78
charbarred	chrisek	CLIFFosakaJAPAN
curtissthompson	DiggityDugged	DigiDave
digitalgopher	dirtyfratboy	drum_bum
EAMUS1CATULI	elebrio	fudgebrown
Gregd	gwjc	IvanB
jasnmb	jefflundberg	johndi
jrepin	kevinrose	koregaonpark
mklopez	motang	MrBabyMan
MrCalifornia	msaleem	ndm007
OBKenobi	Ostermayer	ozguralaz
p9s50W5k4GUD2c6	parislemon	radicaldementia
rodtrent	ryland2	sahaskatta
scoreboard27	seanthebond	sicc
skored	snipehack	starexplorer
supernova17	TheAttacks	TheWalkingDude
titlesaysitall	tomboy501	tommytrc
UCBearcats	webtech	webtickle
zaibatsu	ZaNkY	zepequeno

Table C.3: Usernames of users in the Digg collection.

RSS Feeds	
Politics Top Stories	Business Top Stories
Technology Top Stories	Sports Top Stories
Entertainment Top Stories	Science Top Stories
Most Emailed	Most Emailed Top Stories
Most Emailed U.S.	Most Emailed World
Most Emailed Business	Most Emailed Technology
Most Emailed Oddly Enough	Most Emailed Sports
Most Emailed Health	Most Emailed Science
Most Emailed Op/Ed	Most Emailed Travel
Most Viewed	Most Viewed Top Stories
Most Viewed U.S.	Most Viewed World
Most Viewed Business	Most Viewed Technology
Most Viewed Oddly Enough	Most Viewed Sports
Most Viewed Health	Most Viewed Science
Most Viewed Op/Ed	Most Viewed Travel
Highest Rated	Highest Rated Top Stories
Highest Rated U.S.	Highest Rated World
Highest Rated Business	Highest Rated Technology
Highest Rated Oddly Enough	Highest Rated Sports
Highest Rated Health	Highest Rated Science
Highest Rated Op/Ed	Highest Rated Travel
Top Stories	

Table C.4: RSS Feeds used in the Yahoo! News collection.

APPENDIX D

Statistical Significance Test Results

The following tables show the probability of the actual FMeasure of Classifier X being less than or equal to the actual FMeasure of Classifier Y, after the complete document collection is processed, as determined by the t-distribution. The probabilities for cases, where the observed average FMeasure of Classifier X is less than or equal to the performance of Classifier Y, are omitted.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y			
Classifier X	Rocchio Variant	LMClassifier	MTT	eRocchio
eRocchio	6.5E-05			
LMClassifier	1.8E-02		4.4E-01	4.2E-01
MTT	2.1E-04			3.7E-01
NB(A)	3.5E-12	6.0E-18	1.6E-12	1.2E-09
FSNB(A)	1.2E-12	2.6E-19	1.9E-13	1.1E-10
NB(B)	9.5E-13	2.3E-18	2.6E-14	2.0E-10
NB(C)	9.2E-13	2.4E-18	7.5E-14	2.1E-10
NB(D)	4.6E-13	2.5E-18	6.4E-15	8.6E-11
NB(E)	9.5E-14	1.2E-16	9.2E-17	7.3E-12
FSNB(E)	7.1E-14	1.1E-17	3.1E-16	7.6E-12

Table D.1: Statistical significance test results for the Yahoo! News dataset, Part 1.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	NB(A)	FSNB(A)	NB(B)
FSNB(A)	2.7E-06		4.1E-03
NB(B)	3.7E-03		
NB(C)	1.2E-01		
NB(D)	1.0E-01		
NB(E)	9.9E-04	4.2E-01	2.4E-03
FSNB(E)	2.5E-06	2.8E-02	9.2E-07

Table D.2: Statistical significance test results for the Yahoo! News dataset, Part 2.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	NB(C)	NB(D)	NB(E)
FSNB(A)	2.6E-05	1.8E-03	
NB(B)	2.2E-02	8.7E-02	
NB(D)	2.0E-01		
NB(E)	2.9E-04	4.4E-05	
FSNB(E)	2.1E-08	1.9E-08	1.3E-02

Table D.3: Statistical significance test results for the Yahoo! News dataset, Part 3.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y			
Classifier X	Rocchio Variant	LMClassifier	MTT	eRocchio
Rocchio Variant		1.8E-01	3.1E-02	
LMClassifier			2.8E-01	
eRocchio	2.8E-01	1.4E-01	3.4E-03	
NB(A)	8.0E-02	6.0E-02	3.8E-02	1.1E-01
FSNB(A)	1.3E-02	9.2E-03	7.2E-03	2.0E-02
NB(B)	8.0E-02	6.3E-02	3.4E-02	1.1E-01
NB(C)	5.9E-02	4.2E-02	2.6E-02	8.1E-02
NB(D)	1.2E-01	9.3E-02	4.9E-02	1.4E-01
NB(E)	9.0E-03	9.3E-03	3.4E-03	1.4E-02
FSNB(E)	4.3E-03	3.5E-03	1.9E-03	5.2E-03

Table D.4: Statistical significance test results for the tagger dataset, Part 1.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	NB(A)	FSNB(A)	NB(B)
FSNB(A)	2.3E-02		2.7E-02
NB(B)	4.9E-01		
NB(C)	1.3E-01		2.4E-01
NB(E)	5.4E-02		3.7E-02
FSNB(E)	1.3E-02	1.8E-01	1.1E-02

Table D.5: Statistical significance test results for the tagger dataset, Part 2.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	NB(C)	NB(D)	NB(E)
NB(A)		2.9E-01	
FSNB(A)	3.3E-02	1.8E-02	2.1E-01
NB(B)		2.0E-01	
NB(C)		1.7E-01	
NB(E)	7.6E-02	1.1E-02	
FSNB(E)	1.7E-02	4.3E-03	5.4E-02

Table D.6: Statistical significance test results for the tagger dataset, Part 3.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	Rocchio Variant	LMClassifier	MTT
Rocchio Variant		4.1E-12	1.3E-02
MTT		3.5E-07	
eRocchio	6.9E-08	1.4E-15	7.1E-06
NB(A)		3.0E-11	2.8E-01
FSNB(A)		1.5E-12	3.8E-02
NB(B)		4.9E-12	6.1E-02
NB(C)		1.4E-11	2.0E-02
NB(D)		3.7E-12	1.5E-02
NB(E)	2.2E-01	7.5E-14	3.4E-03
FSNB(E)	2.3E-02	8.4E-16	5.0E-05

Table D.7: Statistical significance test results for the Digg dataset, Part 1.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y		
Classifier X	NB(A)	FSNB(A)	NB(B)
Rocchio Variant	5.2E-03	1.6E-01	9.0E-02
eRocchio	1.4E-08	4.8E-07	5.3E-07
FSNB(A)	6.8E-03		2.0E-01
NB(B)	2.2E-02		
NB(C)	1.7E-05	1.3E-01	1.7E-02
NB(D)	3.4E-03	2.0E-01	2.2E-02
NB(E)	1.3E-04	2.2E-02	9.8E-04
FSNB(E)	3.3E-06	6.0E-05	3.7E-06

Table D.8: Statistical significance test results for the Digg dataset, Part 2.

$P(\bar{F}_X \leq \bar{F}_Y)$	Classifier Y			
Classifier X	NB(C)	NB(D)	NB(E)	FSNB(E)
Rocchio Variant	4.4E-01	4.0E-01		
eRocchio	3.9E-05	1.8E-05	8.2E-03	8.7E-02
NB(C)		4.3E-01		
NB(E)	1.0E-01	6.5E-02		
FSNB(E)	3.1E-03	4.7E-04	2.4E-02	

Table D.9: Statistical significance test results for the Digg dataset, Part 3.

References

[ABP05]   Fabrizio Angiulli, Stefano Basta, and Clara Pizzuti. "Detection and prediction of distance-based outliers." In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pp. 537–542, New York, New York, USA, 2005. ACM Press.

[Ali06]   Alias-I. "LingPipe." `http://www.alias-i.com/lingpipe/index.html`, 9 2006.

[All02]   James Allan. "Detection As Multi-Topic Tracking." *Information Retrieval*, **5**(2-3):139–157, 2002.

[And07]   Chris Anderson. "Calculating latent demand in the long tail." In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1–1, New York, New York, USA, 2007. ACM Press.

[Ang05]   Galia Angelova. *Lecture Notes in Computer Science*, chapter Language Technologies Meet Ontology Acquisition, pp. 367–380. Springer Berlin / Heidelberg, 2005.

[APL98]   James Allan, Ron Papka, and Victor 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*, pp. 37–45, New York, New York, USA, 1998. ACM Press.

[AU06]   Hisham Al-Mubaid and Syed A. Umair. "A New Text Categorization Technique Using Distributional Clustering and Learning Logic." *IEEE Transactions on Knowledge and Data Engineering*, **18**(9):1156–1165, 2006.

[AW06]   Ralitsa Angelova and Gerhard Weikum. "Graph-based text classification: learn from your neighbors." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 485–492, New York, New York, USA, 2006. ACM Press.

[BEL07]   James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk, editors. *Proceedings of KDD Cup and Workshop 2007*. ACM SIGKDD, August 2007.

[BL97]   Avrim L. Blum and Pat Langley. "Selection of relevant features and examples in machine learning." *Artificial Intelligence*, **97**(1-2):245–271, 1997.

[BNJ03]   David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal Machine Learning Res.*, **3**:993–1022, 2003.

[BPC00]   Daniel Billsus, Michael J. Pazzani, and James Chen. "A learning agent for wireless news access." In *IUI '00: Proceedings of the 5th international conference on intelligent user interfaces*, pp. 33–36, New York, New York, USA, 2000. ACM Press.

[Bre96]   Leo Breiman. "Bagging predictors." *Machine Learning*, **24**(2):123–140, 1996.

[Bro02]   Christophe Brouard. "CLIPS at TREC-11: Experiments in Filtering." In *TREC11*, 2002.

[CA06]   Paul J. Chase and Shlomo Argamon. "Stylistic text segmentation." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 633–634, New York, New York, USA, 2006. ACM Press.

[CCG04]   Ricardo Carreira, Jaime M. Crato, Daniel Goncalves, and Joaquim A. Jorge. "Evaluating adaptive user profiles for news classification." In *IUI '04: Proceedings of the 9th international conference on intelligent user interface*, pp. 206–212, New York, New York, USA, 2004. ACM Press.

[CG96]   Stanley F. Chen and Joshua Goodman. "An empirical study of smoothing techniques for language modeling." In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pp. 310–318, Morristown, NJ, USA, 1996. Association for Computational Linguistics.

[CGR05]   Gianna M. Del Corso, Antonio Gulli, and Francesco Romani. "Ranking a stream of news." In *WWW '05: Proceedings of the 14th international conference on the World Wide Web*, pp. 97–106, New York, New York, USA, 2005. ACM Press.

[CH92]   S. le Cessie and J.C. van Houwelingen. "Ridge Estimators in Logistic Regression." *Applied Statistics*, **41**(1):191–201, 1992.

[CSM02]   Amitabh Chaudhary, Alexander S. Szalay, and Andrew W. Moore. "Very Fast Outlier Detection in Large Multidimensional Data Sets." In *Data Mining and Knowledge Discovery*, 2002.

[CVX06]   Paul Chesley, Bruce Vincent, Li Xu, and Rohini K. Srihari. "Using Verbs and Adjectives to Automatically Classify Blog Sentiment." In *Proceedings of the AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, 2006.

[CZ02]   Zhixiang Chen and Binhai Zhu. "Some Formal Analysis of Rocchio's Similarity-Based Relevance Feedback Algorithm." *Information Retrieval*, **5**(1):61–86, 2002.

[Den06]   Peter J. Denning. "Infoglut." *Communcations of the ACM*, **49**(7):15–19, 2006.

[Dig07]   Digg. "Digg." `http://www.digg.com`, September 2007.

[DM06]   Fernando Diaz and Donald Metzler. "Improving the estimation of relevance models using large external corpora." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 154–161, New York, New York, USA, 2006. ACM Press.

[DWK03]   Laurie Damianos, Steve Wohlever, Robyn Kozierok, and Jay Ponte. "MiTAP: A Case Study of Integrated Knowledge Discovery Tools." *hicss*, **03**:69c, 2003.

[ES04]   Roberto Esposito and Lorenza Saitta. "A Monte Carlo analysis of ensemble classification." In *ICML '04: Proceedings of the 21st international conference on machine learning*, p. 34, New York, New York, USA, 2004. ACM Press.

[Esk00]   Eleazar Eskin. "Detecting errors within a corpus using anomaly detection." In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pp. 148–153, San Francisco, California, USA, 2000. Morgan Kaufmann Publishers Inc.

[FGM05]   Jenny Rose Finkel, Trond Grenager, and Christopher Manning. "Incorporating non-local information into information extraction systems by Gibbs sampling." In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370, Morristown, NJ, USA, 2005.

[Fin08]   Chris Finke. "Diggs Top 100 User." `http://www.efinke.com/digg/topusers.html`, July 2008.

[Fog03]   B. J. Fogg. "Prominence-interpretation theory: explaining how people assess credibility online." In *CHI '03: CHI '03 extended abstracts on Human factors in computing systems*, pp. 722–723, New York, New York, USA, 2003. ACM Press.

[For04]   George Forman. "A Pitfall and Solution in Multi-Class Feature Selection for Text Classification." In *Proceedings of the 21st International Conference on machine learning*, p. 38, 2004.

[For06]   George Forman. "Tackling concept drift by temporal inductive transfer." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 252–259, New York, New York, USA, 2006. ACM Press.

[FS96]   Yoav Freund and Robert E. Schapire. "Experiments with a New Boosting Algorithm." In *International Conference on machine learning*, pp. 148–156, 1996.

[FWM01]   Martin Franz, Todd Ward, J. Scott McCarley, and Wei-Jing 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*, pp. 310–317, New York, New York, USA, 2001. ACM Press.

[GE03]   Isabelle Guyon and Andre Elisseeff. "An introduction to variable and feature selection." *Journal of Machine Learning Research*, **3**:1157–1182, 2003.

[GLM04]   Alexander Genkin, David D. Lewis, and David Madigan. "Large-Scale Bayesian Logistic Regression for Text Categorization." *Journal of Machine Learning Research*, 2004.

[GM07]   Evegniy Gabrilovich and Shaul Markovitch. "Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis." In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 1606–1611, 2007.

[Goo08]   Google. "Google News." `http://news.google.com/`, 2008.

[GR08]   Rakesh Gupta and Lev Ratinov. "Text Categorization with Knowledge Transfer from Heterogeneous Data Sources." In *Proceedings of*

the 23rd AAAI Conference on Artificial Intelligence, Chicago, Illinois, July 2008.

[GS04]   T. L. Griffiths and M. Steyvers. "Finding scientific topics." *Proceedings of tha National Academy of Science*, **101 Supplement 1**:5228–5235, April 2004.

[Hay06]   Rick Hayes-Roth. "Two theories of process design for information superiority: Smart pull vs. smart push." In *Proceedings of the command and control research and technology symposium: The state of the art and the state of the practice*, San Diego, California, 2006.

[Hen06]   Monika 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*, pp. 284–291, New York, New York, USA, 2006. ACM Press.

[JL95]   George H. John and Pat Langley. "Estimating Continuous Distributions in Bayesian Classifiers." In *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence*, pp. 338–345, 1995.

[Joa96]   Thorsten Joachims. "A Probablistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization." Technical Report CMU-CS-96-118, Carnegie Mellon University, 1996.

[Kar03]   George Karypis. "CLUTO: a software package for clustering high-dimensional datasets." `http://www-users.cs.umn.edu/~karypis/cluto/index.html`, 2003.

[KJ97]   Ron Kohavi and George H. John. "Wrappers for feature subset selection." *Artificial Intelligence*, **97**(1-2):273–324, 1997.

[KSA06]   Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri. "Authorship attribution with thousands of candidate authors." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 659–660, New York, New York, USA, 2006. ACM Press.

[KZL08]   Jonathan Koren, Yi Zhang, and Xue Liu. "Personalized interactive faceted search." In *WWW '08: Proceeding of the 17th international conference on the World Wide Web*, pp. 477–486, New York, New York, USA, 2008. ACM.

187

[LDD05]   H. Liu, E.R. Dougherty, J.G. Dy, K. Torkkola, E. Tuv, H. Peng, C. Ding, F. Long, M. Berens, L. Parsons, Z. Zhao, L. Yu, and G. Forman. "Evolving feature selection." *IEEE Intelligent Systems*, **20**(6):64–76, November - December 2005.

[LHF05]   Niels Landwehr, Mark Hall, and Eibe Frank. "Logistic Model Trees." *Machine Learning*, **59**(1-2):161–205, 2005.

[Lia04]   Jiu-Zhen Liang. "SVM multi-classifier and Web document classification." In *Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on*, volume 3, pp. 1347–1351, 2004.

[Lit88]   Nick Littlestone. "Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm." *Machine Learning*, **2**(4):285–318, 1988.

[LK05]   Aleksandar Lazarevic and Vipin Kumar. "Feature bagging for outlier detection." In *KDD '05: Proceeding of the 11th ACM SIGKDD international conference on knowledge discovery in data mining*, pp. 157–166, New York, New York, USA, 2005. ACM Press.

[LLK03]   Hung-Jen Lai, Ting-Peng 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*, pp. 225–229, New York, New York, USA, 2003. ACM Press.

[LV02]   Yihua Liao and V. Rao Vemuri. "Using Text Categorization Techniques for Intrusion Detection." In *Proceedings of the 11th USENIX Security Symposium*, pp. 51–59, Berkeley, California, USA, 2002. USENIX Association.

[LY05]   Huan Liu and Lei Yu. "Toward Integrating Feature Selection Algorithms for Classification and Clustering." *IEEE Transactions on Knowledge and Data Engineering*, **17**(4):491–502, 2005. Senior Member-Huan Liu and Student Member-Lei Yu.

[LZC06]   Jiexun Li, Rong Zheng, and Hsinchun Chen. "From fingerprint to writeprint." *Communcations of the ACM*, **49**(4):76–82, 2006.

[MAS04]   Juha Makkonen, Helena Ahonen-Myka, and Marko Salmenkivi. "Simple Semantics in Topic Detection and Tracking." *Information Retrieval*, **7**(3-4):347–368, 2004.

[MCM02]   Liang Ma, Qunxiu Chen, Shaping Ma, Min Zhang, and Lianhong Cai. "Incremental Learning for Profile Training in Adaptive Document Filtering." In *TREC11*, 2002.

[Mis05]   G. Mishne. "Experiments with Mood Classification in Blog Posts." In *Style2005 - the 1st Workshop on Stylistic Analysis Of Text For Information Access, at SIGIR*, 2005.

[MM04]   Prem Melville and Raymond J. Mooney. "Diverse ensembles for active learning." In *ICML '04: Proceedings of the 21st international conference on machine learning*, p. 74, New York, New York, USA, 2004. ACM Press.

[MP01]   Sofus A. Macskassy and Foster Provost. "Intelligent information triage." In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 318–326, New York, New York, USA, 2001. ACM Press.

[MZ05]   Qiaozhu Mei and ChengXiang Zhai. "Discovering evolutionary theme patterns from text: an exploration of temporal text mining." In *KDD '05: Proceeding of the 11th ACM SIGKDD international conference on knowledge discovery in data mining*, pp. 198–207, New York, New York, USA, 2005. ACM Press.

[NCS06]   David Newman, Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. "Analyzing entities and topics in news articles using statistical topic models." In *IEEE International Conference on Intelligence and Security Informatics*, 2006.

[Net07]   Netflix. "Netflix Prize." `http://www.netflixprize.com/`, September 2007.

[NF05]   Petteri Nurmi and Patrik Floreen. "Online feature selection for contextual time series data." In *PASCaliforniaL Subspace, Latent Structure and Feature Selection Workshop*, Bohinj, Slovenia, February 2005.

[NIS04]   NIST. "The Topic Detection and Tracking 2004 (TDT-2004) Evaluation Project." `http://www.nist.gov/speech/tests/tdt/tdt2004/index.htm`, December 2004.

[NSK06]   Ryosuke Nagura, Yohei Seki, Noriko Kando, and Masaki Aono. "A method of rating the credibility of news documents on the web." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 683–684, New York, New York, USA, 2006. ACM Press.

[Ope06]   OpenNLP. "openNLP." `http://opennlp.sourceforge.net/`, 9 2006.

[PBM98]   Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." Technical report, Stanford Digital Library Technologies Project, 1998.

[PC05]   Raymond K. Pon and Alfonso F. Cárdenas. "Data quality inference." In *IQIS '05: Proceedings of the 2nd international workshop on information quality in information systems*, pp. 105–111, New York, New York, USA, 2005. ACM Press.

[PCB07a]   Raymond K. Pon, Alfonso F. Cárdenas, David Buttler, and Terence 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.

[PCB07b]   Raymond K. Pon, Alfonso F. Cárdenas, David Buttler, and Terence 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*, pp. 560–569, New York, New York, USA, 2007. ACM Press.

[PCB08a]   Raymond K. Pon, Alfonso F. Cárdenas, and David J. Buttler. "Improving Naive Bayes with Online Feature Selection for Quick Adaptation to Evolving Feature Usefulness." In *2008 SIAM SDM Text Mining Workshop*, Atlanta, Georgia, April 2008.

[PCB08b]   Raymond K. Pon, Alfonso F. Cárdenas, and David J. Buttler. "Online Selection of Parameters in the Rocchio Algorithm for Identifying Interesting News Articles." In *10th ACM International Workshop on Web Information and Data Management (WIDM)*, Napa Valley, California, October 2008.

[PLT05]   Ramani S. Pilla, Catherine Loader, and Cyrus C. Taylor. "New Technique for Finding Needles in Haystacks: Geometric Approach to Distinguishing between a New Source and Random Fluctuations." *Physical Review Letters*, **95**(23):230202, 2005.

[Por80]   M. F. Porter. "An algorithm for suffix stripping." *Program*, **14**(3):130–137, 1980.

[PSW03]   Fuchun Peng, Dale Schuurmans, and Shaojun Wang. "Language and task independent text categorization with simple language models."

In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pp. 110–117, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[PT03]   Simon Perkins and James Theiler. "Online Feature Selection using Grafting." In *ICML*, pp. 592–599, 2003.

[Qui93]   Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California, 1993.

[RE06]   David L. Roberts and Tina Eliassi-Rad. "A Position Paper: Value of Information for Evidence Detection." In *AAAI Fall Symposium on Capturing and Using Patterns for Evidence Detection*, Arlington, VA 2006.

[RFZ01]   Dragomir Radev, Weiguo Fan, and Zhu Zhang. "WebInEssence: A personalised web-based multi-document summarisation and recommendation system." In *Proceedings of the NAACL-01*, pp. 79–88, 2001.

[RJ05]   Matthew J. Rattigan and David Jensen. "The case for anomalous link detection." In *4th Multi-Relational Data Mining Workshop, 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2005.

[Roc71]   J. Rocchio. *Relevance Feedback in Information Retrieval*, chapter 14, pp. 313–323. Prentice-Hall, 1971.

[RP04]   Ellen Riloff and William Phillips. "An Introduction to the Sundance and AutoSlog Systems." Technical Report UUCS-04-015, University of Utah School of Computing, 2004.

[RS02]   Stephen Robertson and Ian Soboroff. "The TREC 2002 Filtering Track Report." In *TREC 2002*, 2002.

[RWZ02]   S.E. Robertson, S. Walker, H. Zaragoza, and R. Herbrich. "Microsoft Cambridge at TREC 2002: Filtering Track." In *TREC11*, 2002.

[SA06]   Mark D. Smucker and James Allan. "Find-similar: similarity browsing as a search tool." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 461–468, New York, New York, USA, 2006. ACM Press.

[Sin04]   Natasha Singh. *"The Use of Syntactic Structure in Relationship Structure."*. Master's thesis, Massachusetts Institute of Technology, 2004.

[SMB97]   Amit Singhal, Mandar Mitra, and Chris Buckley. "Learning routing queries in a query zone." In *Proceedings of the 20th Annual Internal ACM SIGIR Conference on research and Development in Information Retrieval*, pp. 25–32, July 1997.

[Sol08]   Solr. "Welcome to Solr." `http://lucene.apache.org/solr/`, 2008.

[SSS98]   Robert E. Schapire, Yoram Singer, and Amit Singhal. "Boosting and Rocchio applied to text filtering." In *Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval*, pp. 215–223, Melbourne, AU, 1998. ACM Press, New York, US.

[Ste06]   Mark Steyvers. *Latent Semantic Analysis: A Road to Meaning*, chapter Probabilistic Topic Models. Laurence Erlbaum, 2006.

[TB98]   Fiona J. Tweedie and R. Harald Baayen. "How variable may a constant be? Measures of lexical richness in perspective." *Computers and the Humanities*, **32**:323–352, 1998.

[TG04]   A.L. Turinsky and R.L. Grossman. "A greedy algorithm for selecting models in ensembles." In *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 547–550, November 2004.

[UBC97]   Paul E. Utgoff, Neil C. Berkman, and Jeffrey A. Clouse. "Decision Tree Induction Based on Efficient Tree Restructuring." *Machine Learning*, **29**(1), October 1997.

[UIM08]   Apache UIMA. "Unstructured Information Management Architecture SDK." `http://incubator.apache.org/uima/`, 9 2008.

[WF05]   Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[WHN02] Lide Wu, Xuanjing Huang, Junyu Niu, Yingju Xia, Zhe Feng, and Yaqian Zhou. "FDU at TREC2002: Filtering, Q&A, Web and video tasks." In *TREC11*, 2002.

[Wie00]   Janyce Wiebe. "Learning Subjective Adjectives from Corpora." In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pp. 735–740. AAAI Press / The MIT Press, 2000.

[Wie02]   Janyce Wiebe. "Instructions for Annotating Opinions in Newspaper Articles." Tr-02-101, Department of Computer Science, University of Pittsburgh, 2002.

[Wik08]   Wikipedia. "Wikipedia, the free encyclopedia." `http://en.wikipedia.org/`, 2008.

[WKY96]   Jacqueline W. T. Wong, W. K. Kan, and Gilbert Young. "ACTION: Automatic classification for full-text documents." *SIGIR Forum*, **30**(1):26–41, 1996.

[WS03]   Scott White and Padhraic Smyth. "Algorithms for estimating relative importance in networks." In *KDD '03: Proceedings of the 9th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 266–275, New York, New York, USA, 2003. ACM Press.

[WVR06]   Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. "Unifying user-based and item-based collaborative filtering approaches by similarity fusion." In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 501–508, New York, New York, USA, 2006. ACM Press.

[WWB04]   Janyce Wiebe, Theresa Wilson, Rebecca Bruce, Matthew Bell, and Melanie Martin. "Learning Subjective Language." *Computational Linguistics*, **30**(3):277–308, 2004.

[XJK01]   E.P. Xing, M.I. Jordan, and R.M. Karp. "Feature selection for high-dimensional genomic microarray data." In *Proceedings of the Eighteenth International Conference on machine Learning*, 2001.

[XYW02]   Hongbo Xu, Zhifeng Yang, Bin Wang, Bin Liu, Jun Cheng, Yue Liu, Zhe Yang, Xueqi Cheng, and Shuo Bai. "TREC-11 Experiments at CaliforniaS-ICT: Filtering and Web." In *TREC11*, 2002.

[Yah07]   Yahoo! "Yahoo! News RSS Feeds." `http://news.yahoo.com/rss`, 2007.

[YH06]   Rong Yan and Alexander 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*, pp. 324–331, New York, New York, USA, 2006. ACM Press.

[YP97]   Yiming Yang and Jan O. Pedersen. "A Comparative Study on Feature Selection in Text Categorization." In *ICML '97: Proceedings of the 14th International Conference on machine learning*, pp. 412–420, San Francisco, California, USA, 1997. Morgan Kaufmann Publishers Inc.

[YZH03]   Hwanjo Yu, ChengXiang Zhai, and Jiawei Han. "Text classification from positive and unlabeled documents." In *CIKM '03: Proceedings of the 12th international conference on Information and knowledge management*, pp. 232–239, New York, New York, USA, 2003. ACM Press.