An Investigation of Active Learning Techniques for Restaurant Recommendations

Alen Lukic
Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
`alukic@cs.cmu.edu`

Abstract

This paper investigates the use of active learning in a restaurant recommendation system. The goal of the system is to attempt to learn the user's preferences over time using an ensemble of machine learning techniques. Because of time constraints, obtaining real user feedback over a period of time proved infeasible. Instead, I perform simulated feedback experiments which mimic a user's incremental feedback to the system. Generally, the system performs well, but its performance is highly dependent on how dicriminative the user's preferences are.

1 Introduction

Recommender systems are systems which generally utilize machine learning in order to predict a user's preferences for items based on existing data. Such systems have varied commercial applications; one famous example is the Netflix Prize competition, in which Netflix awarded a prize of $1,000,000 to a team which built a recommender system superior to Netflix's own [1].

Currently, there is no well-known commercial system for providing users with restaurant recommendations. The goal of this paper is to investigate whether building such a system would be feasible based on system performance in experiments which simulate real user feedback over time. This is a form of active learning [2].

2 Related Work

He, et. al. explore active collaborative filtering for solving the cold start problem (that is, making recommendations to users before having sufficient information about their preferences) [3]. In this work the researchers actively obtain feedback from users prior to attempting to provide them with recommendations. Boutilier, et. al. also investigate active recommender systems from the perspective of defining information gain from user feedback observations in order to optimize computation time [4]. Additionally, many restaurant recommender system prototypes can be found on the Internet; Gupta and Singh's proposed system somewhat resembles the one discussed in this paper [5].

This system bears some similarity to that of He, et. al. in that a user's preferences are derived by querying the user with a restaurant profile and a picture of the type of cuisine served there. The user then indicates whether they would eat at the restaurant. This information is used in order to provide the user with recommendations.

Because this system maintains a separate model for each user, and the training data for each user is relatively small (even in large cities like New York, the number of possible training instances would still be limited to the thousands), the system does not explicitly consider the information gain from new observations, as per the work by Boutilier, et. al. The computational expense of doing so would actually significantly outweigh any performance gain from choosing not to add an observation to the training data, because the training data for any given user is bounded by the number of restaurants in the user's vicinity.

3 Methodology

3.1 Data Collection

In order to collect restaurant data, I made use of the Yelp Search API [6]. Starting with an arbitrary restaurant address near the vicinity of downtown Pittsburgh, I queried the Yelp API in a breadth first search-like manner; the restaurant network can be represented as a graph $G = \{V, E\}$, where nodes $\forall r \in V$ are restaurants and edges $\{v_i, v_j\} \in E$ represent connected nodes. Restaurants are connected if one restaurant can be found by querying the API with the term "restaurant" using the address of the other restaurant as the location, and vice-versa. Using this algorithm, I collected data for 1,000 restaurants in Pittsburgh and the surrounding areas. Some of the businesses crawled were not actually restaurants, but these represent a very small fraction of the data.

To collect user observations for use in experiments, I built a GUI which displayed restaurant details to users. The user then indicated whether they would eat at this restaurant. This GUI was deployed and used with two human subjects, each of whom cast a vote for all 1,000 restaurants.

3.2 Data Representation

The system trained an explicit model for each user's taste preferences. The training observations were of the following form, containing the restaurant's average rating, number of reviews, binary indicators of the 113 detected cuisine types, and a label indicating whether the user would eat at this restaurant.

$$\mathbf{x} = \{x_r, x_{rc}, x_0, \ldots, x_{112}\}, \mathbf{y} = \{0, 1\}$$

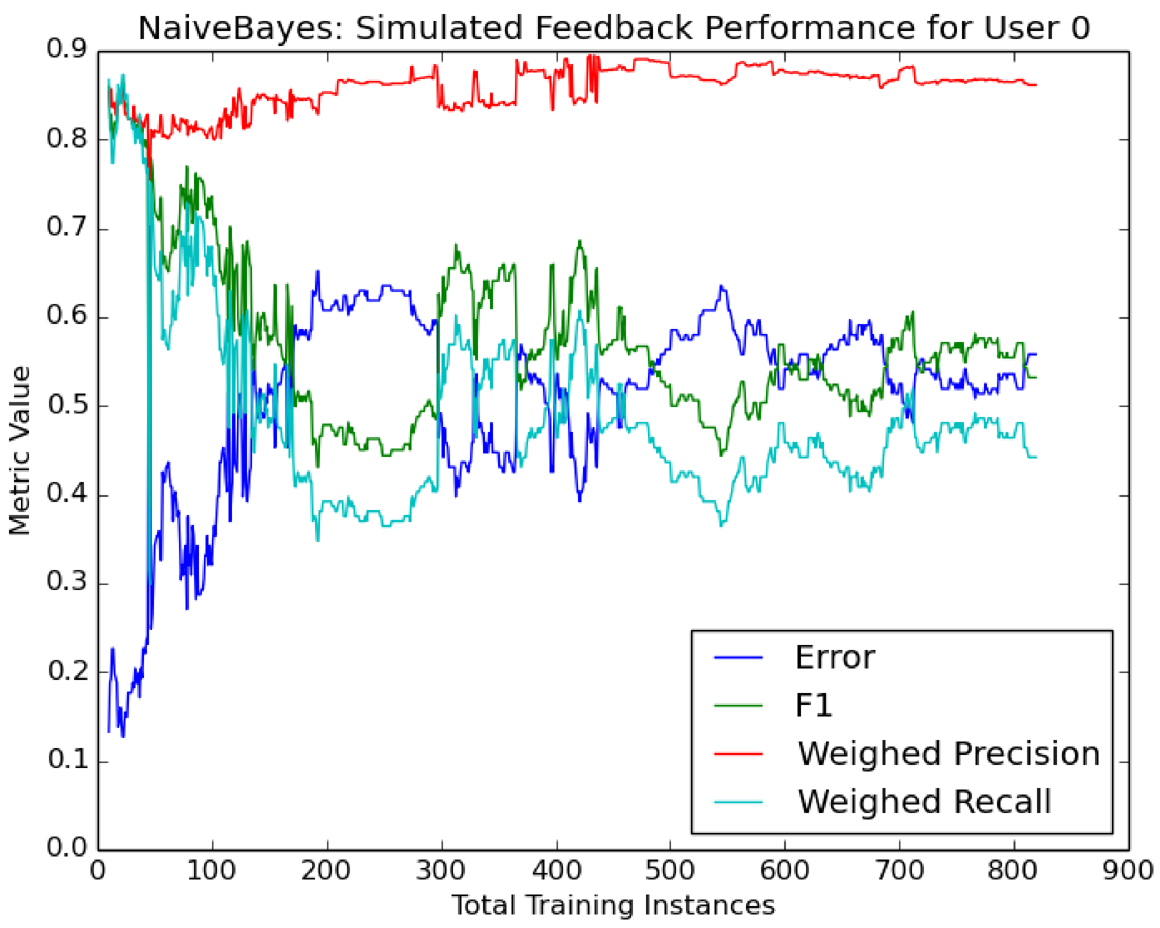
3.3 User Models

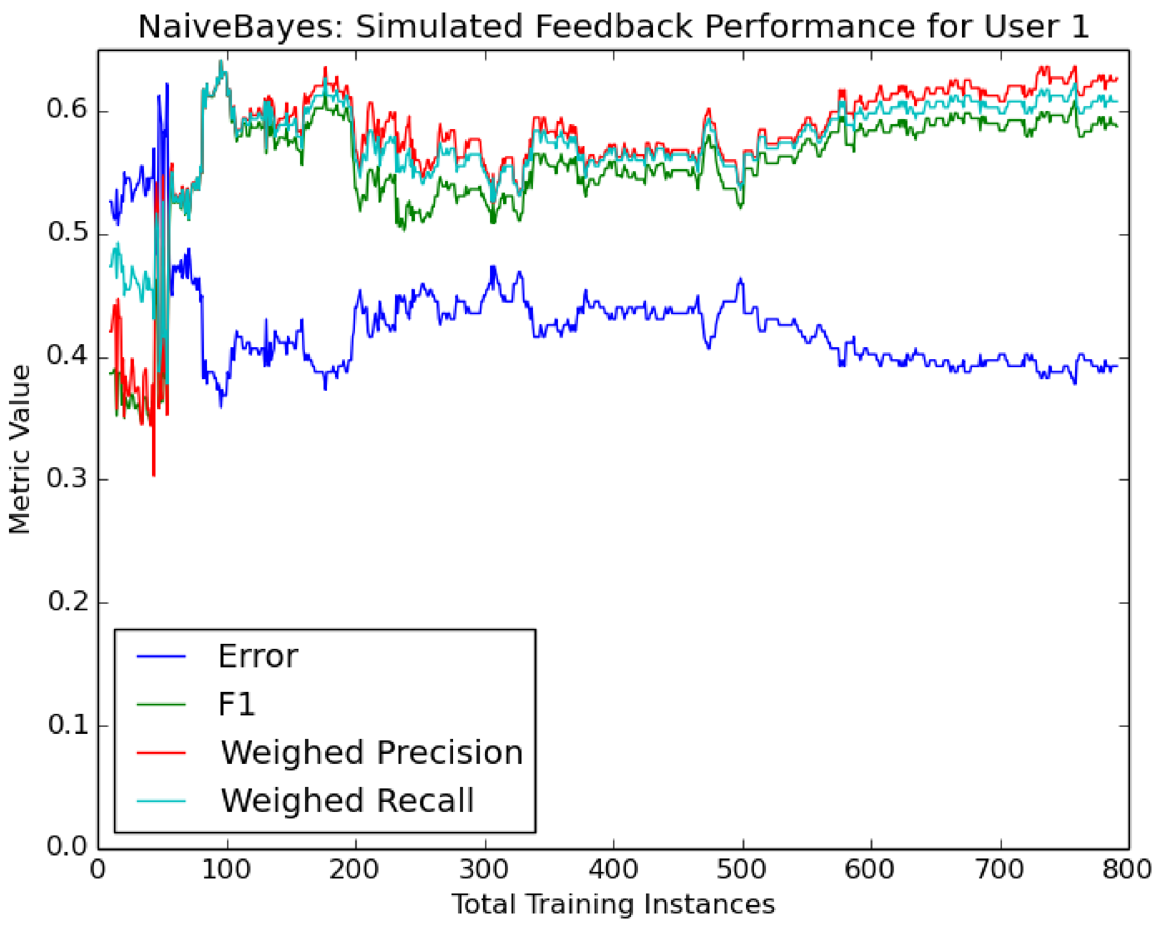
The Weka machine learning toolkit was used in order to efficiently explore different machine learning models for user taste profiles [7]. Because the purpose of this system is to recommend new restaurants to users, it is sufficient for the system to make a binary prediction about whether a particular user would want to eat a particular restaurant. Hence, on this iteration of the system, the user models were produced using binary classifiers. Several machine learning algorithms were explored. Weka has built-in cross-validation and parameter-tuning functionality, which was also utilzed in order to optimize the trained models.

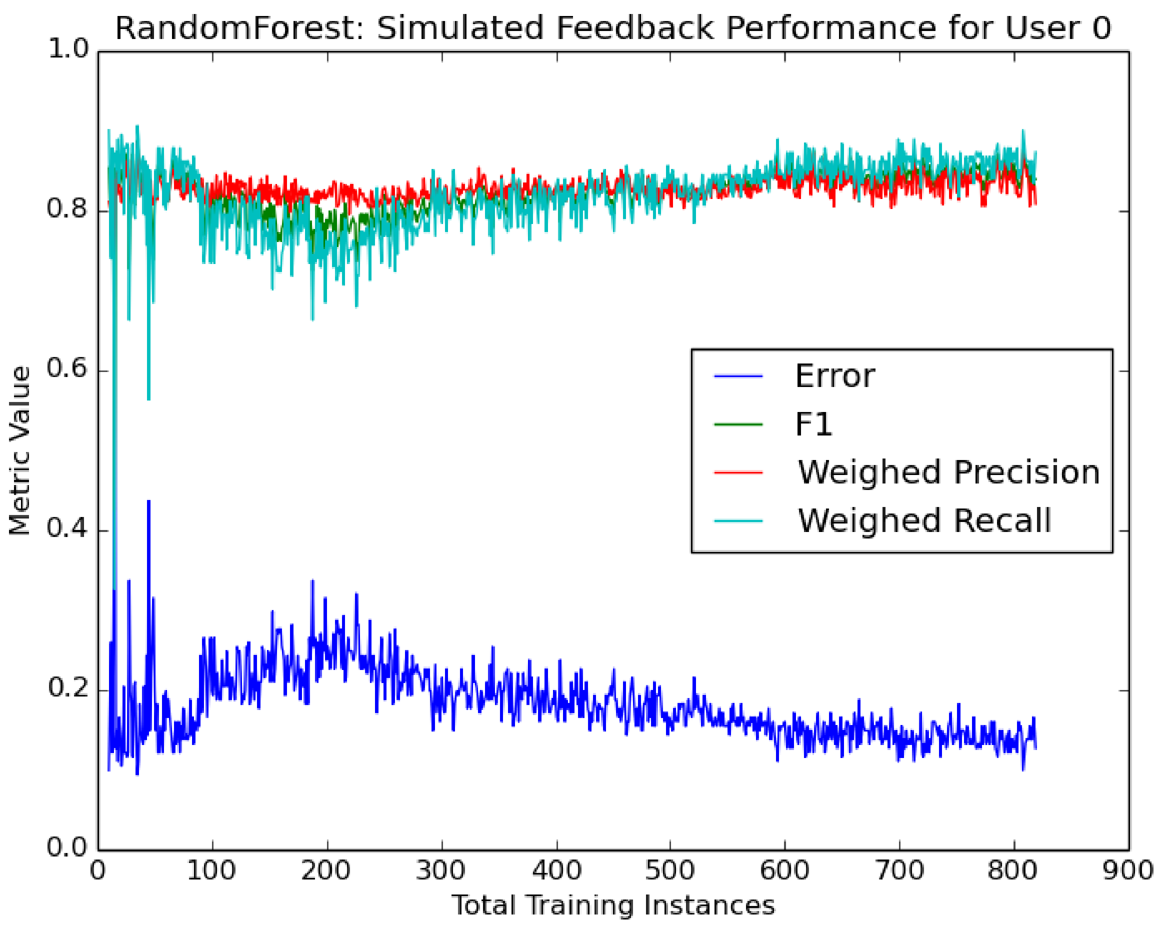
4 Experiments

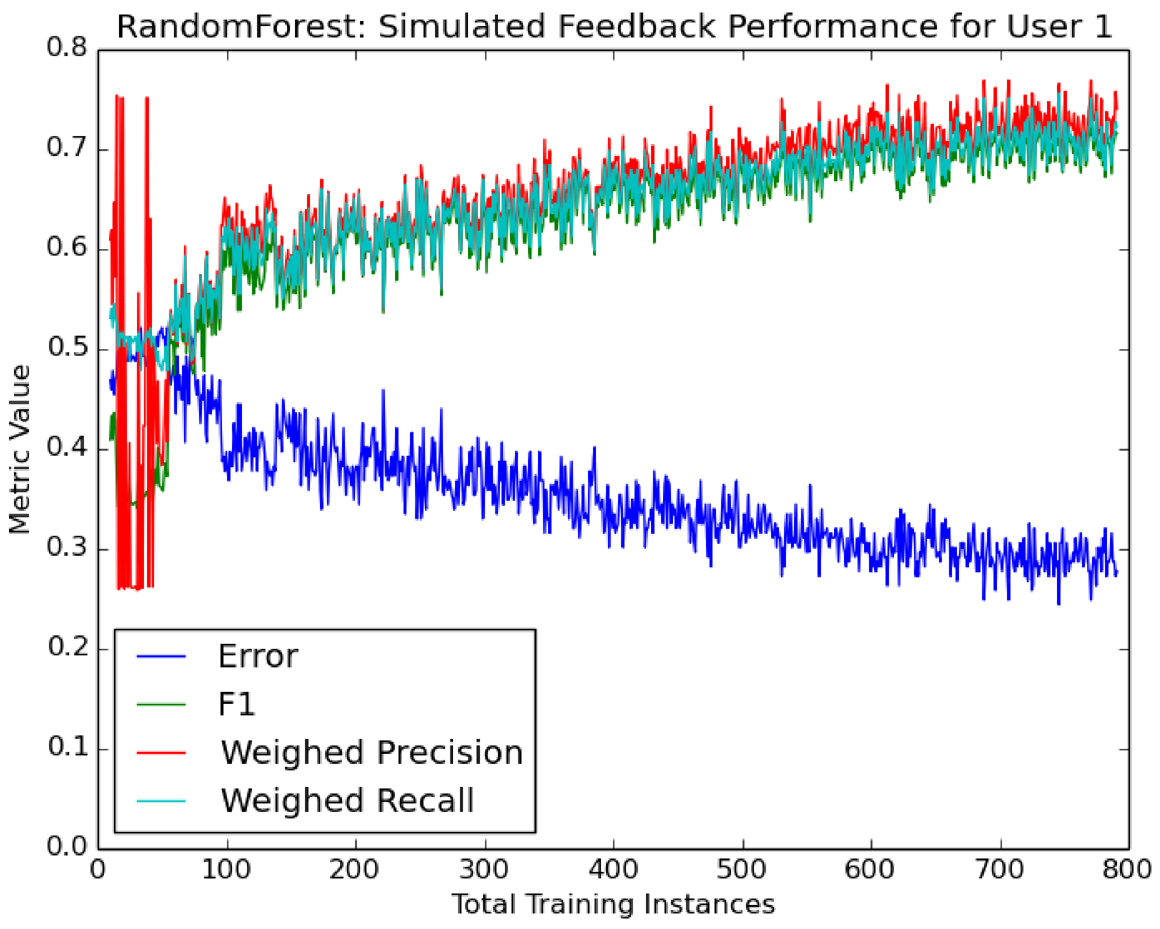
4.1 Simulated Feedback Experiments

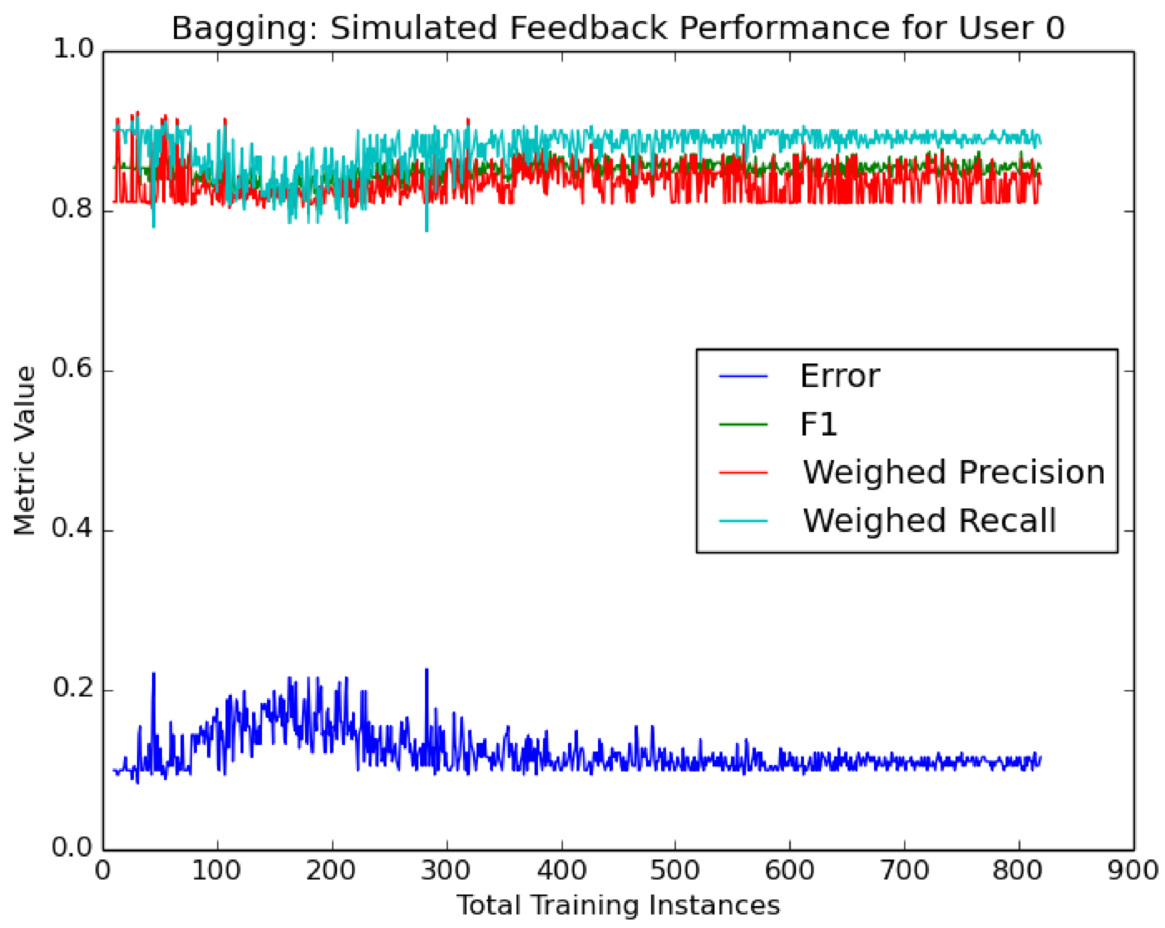
Each user's feedback data was randomly split into 80/20 train/test sets. Then, each Weka model was trained iteratively, starting with only 10 training instances and adding one per iteration, up to the cardinality of the training set. This procedure simulates progressively receiving user feedback over time. The following graphs show the performance metrics of each model as the number of training instances grows. The naive Bayes model is used as a baseline.

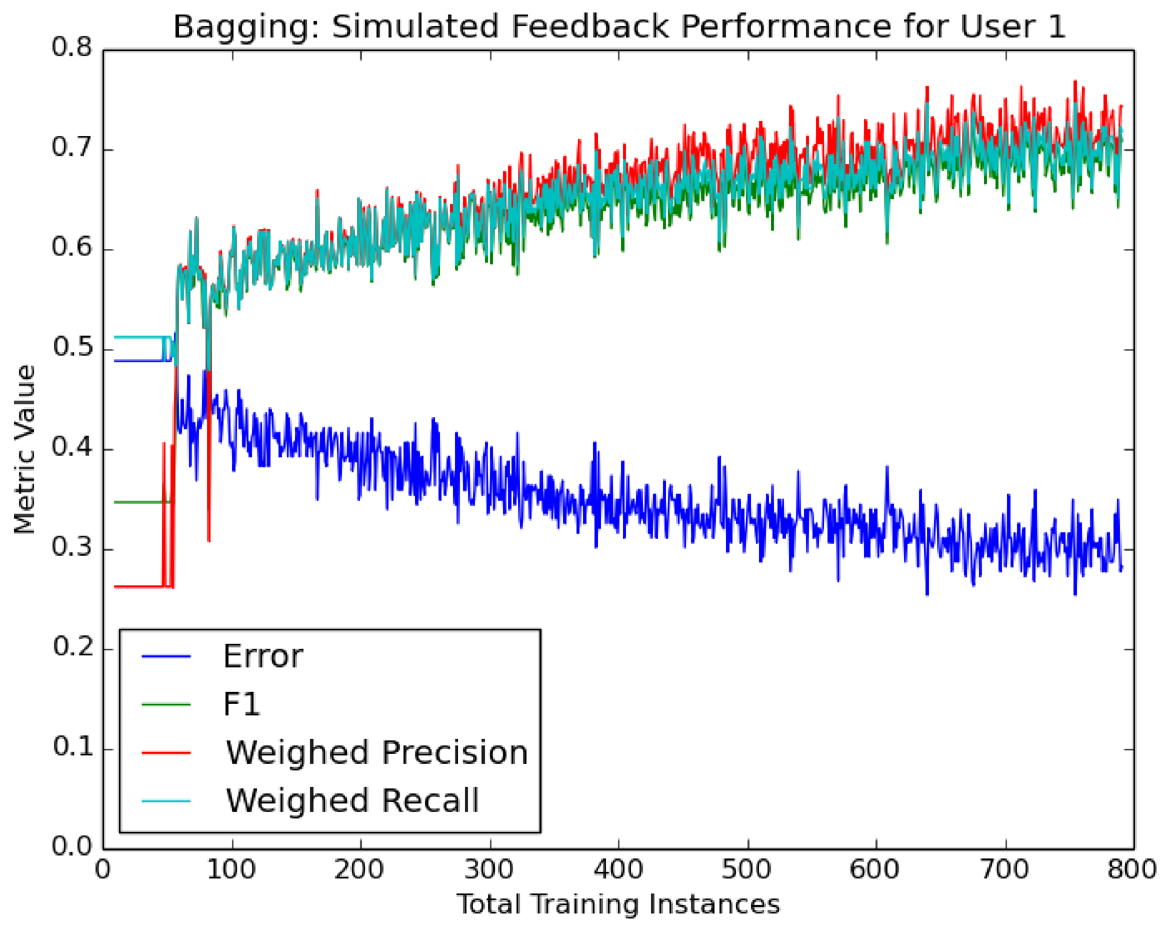

NaiveBayes: Simulated Feedback Performance for User 0
Metric Value
Total Training Instances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
0
100
200
300
400
500
600
700
800
900
Error
F1
Weighed Precision
Weighed Recall


NaiveBayes: Simulated Feedback Performance for User 1
Metric Value
Total Training Instances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
100
200
300
400
500
600
700
800
Error
F1
Weighed Precision
Weighed Recall


RandomForest: Simulated Feedback Performance for User 0
Metric Value
Total Training Instances
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600
700
800
900
Error
F1
Weighed Precision
Weighed Recall


RandomForest: Simulated Feedback Performance for User 1
Metric Value
Total Training Instances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
100
200
300
400
500
600
700
800
Error
F1
Weighed Precision
Weighed Recall


Bagging: Simulated Feedback Performance for User 0
Metric Value
Total Training Instances
0.0
0.2
0.4
0.6
0.8
1.0
0
100
200
300
400
500
600
700
800
900
Error
F1
Weighed Precision
Weighed Recall


Bagging: Simulated Feedback Performance for User 1
Metric Value
Total Training Instances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
100
200
300
400
500
600
700
800
Error
F1
Weighed Precision
Weighed Recall

The following is a tabular version of the classifiers' performance when each users' full training set was used. The entries are formatted as user0 / user1.

Classifier	*Error Rate*	*Weighed Precision*	*Weighed Recall*	F_1
Naive Bayes	0.392 / 0.558	0.623 / **0.862**	0.608 / 0.442	0.587 / 0.532
Random Forest	**0.278** / 0.127	0.741 / 0.808	**0.722** / 0.873	**0.716** / 0.839
Bagging	0.282 / **0.116**	**0.743** / 0.833	0.718 / **0.884**	0.708 / **0.853**

4.2 Analysis

There is a noteworthy improvement in the harmonic F_1 mean over the baseline naive Bayes classifier when more complex models are trained, and in general, the model metrics of the random forest and bootstrap aggregation classifiers are superior to that of naive Bayes. Of all classifiers explored, these two classifiers produced, on average, the best models for each of the two human users.

What is most interesting is the characteristic differences between the models produced for the users for all classifiers. That is, generally speaking, user 1's model tends to improve much more than user 2's model on a relative scale when the classifier has more training data to ingest. This is particularly evident in the case of the bootstrap aggregation classifier, where user 0's performance metrics are virtually flat regardless of the number of training instances, whereas user 1's performance metrics improve linearly with more training data.

These results imply that no single model will be optimal for all system users. Also, the performance of any classifier will be determined by how discriminative the user's training data is; that is, it will be difficult for a model to make quality predictions if a user's preference distribution is relatively flat. This is supported by the fact that, in this experiment, user 0 voted positively for 53% of the restaurants, whereas user 1 voted positively only for 13% of the restaurants. User 1 was much more discriminative.

5 Conclusions

Based on these experiments, a restaurant recommendation system which uses active learning to learn a user's preferences over time is feasible. Through the use of the Yelp Search API or a similar service, it is extremely easy to collect significant amounts of quality data about restaurants in a particular location. The results indicate that models can be trained to predict which restaurants a user will like with both high precision and recall and an accuracy which is significantly better than chance, despite the varying quality of models caused by discriminative differences between users.

6 Future Work

The results in this paper indicate that model selection and optimization may be a challenge. Exploring collaborative filtering rather than explicit user models may prove fruitful in addressing this challenge. Furthermore, the quality of the results can be improved by obtaining feedback from more users and from users in different locations. Deploying a service which utilizes this recommender system as its back-end on a mobile platform would allow for more data collection and evaluating the system in a real, rather than simulated, environment.

References

[1] Netflix Prize. http://en.wikipedia.org/wiki/Netflix_Prize

[2] Active learning. http://en.wikipedia.org/wiki/Active_learning_(machine_learning)

[3] He, L.; Liu, N.; & Yang, Q. Active Dual Collaborative Filtering with both Item and Attribute Feedback. *In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.*

[4] Boutilier, C.; Zemel, R.; & Marlin, B. Active Collaborative Filtering. *In Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence, 2003.*

[5] Gupta, A. & Singh, K. Location Based Personalized Recommendation System for Mobile Environments. http://www.academia.edu/4413692/

[6] Yelp Search API. http://www.yelp.com/developers/documentation/v2/search_api

[7] Weka. http://www.cs.waikato.ac.nz/ml/weka/