Context-aware Movie Recommendation based on Signal Processing and Machine Learning

Claudio Biancalana
Department of Computer Science and Automation
ROMA TRE University
Via della Vasca Navale 79
Rome, Italy 00146
claudio.biancalana@dia.uniroma3.it

Fabio Gasparetti
Department of Computer Science and Automation
ROMA TRE University
Via della Vasca Navale 79
Rome, Italy 00146
gaspare@dia.uniroma3.it

Alessandro Micarelli
Department of Computer Science and Automation
ROMA TRE University
Via della Vasca Navale 79
Rome, Italy 00146
micarel@dia.uniroma3.it

Alfonso Miola
Department of Computer Science and Automation
ROMA TRE University
Via della Vasca Navale 79
Rome, Italy 00146
miola@dia.uniroma3.it

Giuseppe Sansonetti
Department of Computer Science and Automation
ROMA TRE University
Via della Vasca Navale 79
Rome, Italy 00146
gsansone@dia.uniroma3.it

ABSTRACT

Most of the existing recommendation engines do not take into consideration contextual information for suggesting interesting items to users. Features such as time, location, or weather, may affect the user preferences for a particular item.

In this paper, we propose two different context-aware approaches for the movie recommendation task. The first is an hybrid recommender that assesses available contextual factors related to time in order to increase the performance of traditional CF approaches. The second approach aims at identifying users in a household that submitted a given rating. This latter approach is based on machine learning techniques, namely, neural networks and majority voting classifiers.

The effectiveness of both the approaches has been experimentally validated using several evaluation metrics and a large dataset.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Information Filtering*

General Terms

Algorithms, Performance, Experimentation

Keywords

Recommender systems, collaborative filtering, context-aware recommendation, signal processing, movie recommendation, neural network, majority voting

1. INTRODUCTION

Recommender systems provide suggestions based on user preferences in order to recommend items likely to be of interest to a user. It is obvious that user preferences are influenced by the current context, such as the time of the day, mood, or the current activity. Nevertheless, few recommender systems explicitly include this information in the preference models. Both context-aware systems and recommender systems are used to provide users with relevant information: the former harnessing user contexts, the latter by means of user interests. It is possible to imagine to combine these two systems. Context-aware recommender systems aim at improving user satisfaction providing better suggestion according with a particular context of use.

Tourist applications and location-based services are two popular examples where context-aware recommendations may improve traditional content-based approaches, such as search engines. While results from information retrieval systems are interpreted as a match to the user query, a recommender system suggests items worthy of consideration.

A special group of recommender systems are the ones based on the collaborative approach [11, 12, 4]. The system generates recommendations using only information about rating profiles for different users. Collaborative systems locate peer users with a rating history similar to the current user and generate recommendations using this neighborhood.

Collaborative filtering (CF) systems have been successful in research, with several projects dealing with the movie recommendation task. The availability of large datasets and additional information that is easy to collect from the web, makes this task interesting.

In this paper, we propose two different approaches for this task. The first is an hybrid recommender that assesses available contextual factors related to time in order to increase the performance of traditional CF approaches. The second approach aims at identifying users that submitted a given rating. This latter approach is based on machine learning techniques, namely, neural networks and majority voting classifiers.

The rest of this paper is organized as follows. Section 2 briefly introduces some related studies on context-aware recommendations. Section 3 details our proposed approaches. The experimental evaluation carried out on the dataset provided by the *2nd Challenge on Context-aware Movie Recommendation* is described in Section 4, while conclusions and future work are presented in the last section.

2. RELATED WORK

There exists a vast literature on recommenders systems (RS). These systems have started to go out from academies and begin to be massively exploited in the everyday life. However, most current recommendations approaches do not exploit additional context data, such as location, mood, or time, in order to improve the accuracy of predictions. Taking advantage of the current location of the user, his companions, and the availability of resources in his surroundings can considerably increase the user satisfaction.

In particular, time is a critical information to analyze in RS. Several works have been proposed on how to exploit temporal information. One of the most relevant contributions on using the temporal context in the recommendations process is described in [7]. The author proposes an approach for modeling the temporal dynamics of customer preferences along the whole time period, in order to separate transient factors from lasting ones. The evolution of an average item rating is considered by a time-dependent and user-independent term. Experimental results show how the inclusion of temporal dynamics can significantly improve the quality of the recommendations.

In [5], the authors advance an extension of the work in [7], in order to make predictions also on the future item popularity based on an estimated polynomial trend.

In the approach presented in Section 3.2 we leverage the information related to the timestamps of ratings. The use of timestamps in the recommendation process has also been advanced in [1]. The authors propose a contextual pre-filtering technique based on implicit user feedback. Specifically, the user profile is split into several sub-profiles, called *micro-profiles*, each representing the user in one particular context. The system proposes recommendations based on these micro-profiles instead of a single user model. The experimental evaluation has been performed on a time-aware music recommendation system.

Other interesting works on the use of temporal information are [6], in which a time weighting scheme for a similarity-based collaborative filtering approach is advanced, and [9], where the authors propose an increasing weight scheme depending on the information recency.

A location-based service able to take into account temporal context is proposed in [3].

3. MOVIE RECOMMENDATION

Recommender engines usually exploit explicit feedbacks (i.e., ratings) or implicit usage data (e.g., if the user has downloaded the movie X) to determine which films the user might want to see next.

Some of the most popular and effective movie recommendation engines are based on collaborative approaches. Collaborative filtering differentiates itself from content-based and demographic recommendations because the content of the suggested items, such as the film genre or its description, or features or profiles of users, such as age and education, are ignored. Without directly interpretable features, most prediction algorithms rely on user or movie similarities to generate predictions.

The traditional user-based CF approach relies on similar users which have similar rating patterns, that is, the prediction of a rating $r_{u,s}$ by user u for movie s is evaluated as an aggregate of the rating of some other users for the same item s. We call these similar users *neighbors*. If a user v is similar to a user u, we say that v is a *neighbor* of u. User-based algorithms generate a prediction for a movie s by analyzing ratings for s from users in u's neighborhood.

In order to draw the distance (or similarity) between two users, the Pearson correlation coefficient is usually employed [11]:

$$sim(u,v) = \frac{\sum_{s \in S_{u,v}} (r_{u,s} - \overline{r}_u)(r_{v,s} - \overline{r}_v)}{\sqrt{\sum_{s \in S_{u,v}} (r_{u,s} - \overline{r}_u)^2 \sum_{s \in S_{u,v}} (r_{v,s} - \overline{r}_v)^2}} \quad (1)$$

where $S_{u,v}$ denotes the set of co-rated items between u and v, $r_{u,s}$ is the rating of the user u for the item s, and $\overline{r}_u$ is the average of the ratings of the user u.

Pearson correlation ranges from 1.0 for users with perfect agreement to -1.0 for users with perfect disagreement. In this way, it is possible to generate a prediction of rating for the user u and the item s as follows:

$$pred(u,s) = \overline{r}_u + \frac{\sum_{v \in NNu} sim(u,v)(r_{v,s} - \overline{r}_v)}{\sum_{v \in NNu} sim(u,v)} \quad (2)$$

where NNu is the set of users in the u's neighborhood.

Our goal is to increase the performance of traditional collaborative filtering approaches by analyzing any contextual factors that are included in the data collected during the normal usage of the recommender system. Traditional CF systems estimate the prediction function of $(user, movie)$ pairs that have not been rated:

$$R : User \times Movie \rightarrow Rating$$

obtaining the top N highest-rated movies. Context-aware collaborative filtering extends the parameters for prediction by adding any available contextual factor:

$$R : User \times Movie \times Context \rightarrow Rating$$

In our case, we have information about the time a particular rating has been submitted by the user. Of course, the timestamp is not directly associated with the time the user watched the rated movie, but in all likelihood the user watched it in a short period of time before the rating submission (e.g., some days or one week). One more contextual feature is the household the user belongs to. Online movie rental and download services such as Netflix[1] and iTunes [2] are becoming very popular. Sometimes the same movie is watched by more members of the same family. Recommender engines should represent and take into account different preferences and suggest movies trying to satisfy all the people of one household at the same time.

In the following sections we introduce two approaches for movie recommenders. The former is able to improve traditional CF by taking into consideration contextual factors such as the timestamp associated to a submitted rating. The second approach, which is based on machine learning techniques, can predict the user that submitted a given rating. Both of the tasks are part of the 2nd Challenge on Context-aware Movie Recommendation[3].

3.1 Movie Recommendation based on Signal Processing

One hypothesis related to the current recommendation task implies that events taking place on a certain moment potentially influence the watched movies. In a preliminary investigation of the dataset used during the evaluation (see Sect. 4), we were able to recognize some interesting characteristics. The most relevant is the amount of ratings that the user usually submits in a given period. Most of the time users gather together a certain number of preferences and submit them to the system in a short period of time, typically around one hour. A very small subset of users spread ratings through several days or weeks. Our hypothesis states that, if a user is particularly interested in watching movies in a particular period, the most watched movies in that period are the ones that should gain more significance during the traditional collaborative recommendation.

For this reason, a pre-processing of the available dataset was performed by grouping together the number of ratings from a given user according with predefined time interval, namely, one day, one week or one month. We obtained quantized samples composing a digital representation of a time-varying quantity, that is, a signal. The first measure of the signal corresponds to the number of movies viewed by the given user in the considered interval. The second measure refers to the subsequent interval, and so on. The same process affected the number of views of a movie m.

Then, it is possible to correlate the two signals drawing similarity measures. In particular, we have signals referring to the user behavior (i.e., watched movies) and a signal associated to each movie. Our goal is to draw a measure of the similarity between the user and a given movie.

In our recommender the cross-correlation of pairs of discrete functions has been employed, which is defined as:

$$(f \star g)_n = \sum_i f_i^* g_{i+n} \tag{3}$$

where f^* is the complex conjugate of f.

The cross-correlation between two input signals is a kind of template matching. The obtained measure is a coefficient of the size and direction of the linear relationship between the functions f and g. If these two signals move together, where they both rise at an identical rate we have positive correlation. If the coefficient is less than 0, we have negative correlation.

The cross-correlation of two input signals is computed by a sample-shift along one of the input signals that is represented by the value of n. In our recommender we have $n = 1$, trying to increase the precision of matching by minimizing the delay of the comparison during the cross correlation assessment.

As for the baseline recommender we employ a user-based CF with a Pearson correlation similarity. Recommendations are produced by finding a neighborhood of N_u similar users. The results of the CF recommendation along with their scores are evaluated by a rescorer. Basically, the results are re-ranked according to the algorithm implemented in the rescorer, which assigns a new score to the items suggested by the traditional CF approach.

In our system, the rescorer computes the cross correlation between the signal associated with the user and the available movies. The first N_m movies ranked by the rescorer are matched with the ones obtained by the CF approach. The movies that are included in both of the result sets are *boosted* with a constant factor $b > 1$ empirically defined with preliminary evaluations.

3.2 Neural Networks and Majority Voting for User Rating Identification

The aim of the second track is to pinpoint which member of a household gave a specific rating. The solution we propose relies on the identification of three simple classifiers c_1, c_2, c_3 that are then combined through a machine learning approach. The problem can be formulated as follows. Given the tuple t:

$$< h, s, r, ts > \tag{4}$$

representing the rating r, given within the household h to the movie s, and with timestamp ts, the goal is to learn a function g that, given the previous tuple, identifies the user u who performed that assessment.

We make the assumption that the users belonging to a given household are known. The first step is to associate those users with the set of *Candidate Users*. Among such users some similarity measures $d(u, t)$ must be adopted in order to identify the nearest one in respect to the submitted tuple 4. We define the following measures, which take into account the parameters of the tuple separately.

Analysis of the distribution of values given by a user (classifier c_1)

We aggregate the frequency of user ratings dividing them into five classes:

1. 0-40
2. 41-60
3. 61-80
4. 81-90
5. 91-100

The aim of such analysis is to model each user based on the values of his ratings. Each user is thus represented by one distribution normalized by an index of Z-score normalization that is compared with the given rating.

[1] www.netflix.com
[2] http://www.apple.com/itunes/
[3] 2011.camrachallenge.com

Analysis of the distribution of times when the rating was given by a user (classifier c_2)

Similarly to the previous case, the input data is processed with the following four groups:

1. morning (7am to 12pm)
2. afternoon (1pm to 5pm)
3. evening (6pm to 10pm)
4. night (11pm to 6am)

This model represents the user habit of giving his ratings at certain times [8]. The values we obtain are then compared with the timestamp ts that appears in the tuples.

Analysis of movies to determine if two users have seen the same movie (classifier c_3)

We analyze the users who have seen a movie. They are compared with the set of *Candidate Users* through the following Jaccard distance:

$$\frac{\cap(S_u, S_v)}{\cup(S_u, S_v)}$$

where S_u and S_v are the set of movies rated by the user u and v, respectively.

Subsequently, the three formulations are combined through a stacking algorithm [2, 10], where a meta-learner based on neural networks derives the best combination of output of the base learners. Details of the architecture of the neural network are shown in Figure 1.

The neural network is trained using the following input parameters (68 features):

1. the distribution of the number of users who rated a movie per week (53 features);
2. the distribution of the number of users who rated a movie per day of the week (7 features);
3. the distribution of ratings given to a movie, divided in five groups (5 features);
4. the submission date of the rating, identified by week of the year (from 1 to 53) and by day of the week (from 1 to 7) (2 features);
5. the number of ratings given to a movie (1 feature).

The output layer of the network consists of three nodes which express a membership value to the three simple classifiers mentioned above.

Finally, the predicted user is chosen through a weighted stacking algorithm according to the following algorithm:

1. `Given the training data` $(\mathtt{t}_1, \vec{c_1}), ..., (\mathtt{t}_N, \vec{c_N})$ `and a tuple` t
2. `For` $i = 1$ `to 3:`
   (a) $TS_i \leftarrow$ `a selection of` N `random examples from the training set`
   (b) $f_i \leftarrow$ `the result of training base learning algorithm on` TS_i
   (c) `draw the output` $y_{c_1}, y_{c_2}, y_{c_3}$ `from the neural network`
3. `Output from the combined classifier:` $g(t)$ `= majority`$(y_{c_1} \cdot f_1(t), y_{c_2} \cdot f_2(t), y_{c_3} \cdot f_3(t))$

where t_i is the tuple and $\vec{c_i}$ is a 3-bit code word vector, which represents the classification of c_1, c_2, and c_3 for the tuple t_i. For example, the vector $[0, 1, 1]$ describes the situation in which the tuple is classified correctly only by the classifiers c_2 and c_3.

The neural network, thus, provides a probability value (i.e., between 0 and 1), which is used as a weight in the majority voting that is defined by the function $g(t)$.

4. EVALUATION

The evaluation is based on the dataset provided for the 2nd Challenge on Context-aware Movie Recommendation. The training set is composed of 4536891 ratings from 171670 different users over a range of 23974 movies. The households are 286 and the users associated with them are 602. Ratings are integers ranging from 0 (meaning "dislike") to 100 (meaning "like").

The CAMRa recommendation tracks address the contextual temporal dimension in the movie recommendation domain. The dataset contains two kinds of contextual features: households and timestamps. On average, we have 2.07 users per household. Timestamps represent the moment when a generic rating has been submitted by one user. An undisclosed offset affects the actual time of all the timestamps.

The challenge focuses on classification accuracy metrics for two different tracks as follows:

Track 1 It is requested to generate recommendations for each single household. The number of recommendations to suggest is predetermined for each household.

Track 2 In this track the goal is to identify which member of a household performed a given rating. Of course, the members of the households are known.

The dataset is anonymized so there is no information about one user besides his household and the submitted ratings. The challenge does not allow participants to take advantage of external sources such as Internet Movie Data Base (IMDB)[4], Wikipedia[5], or NetFlix Web sites and no additional information about the user is provided. Therefore, demographic recommendation based, for example, on features like age, gender, profession, income, or location, is not feasible.

We evaluated the performance of our approaches through the following well-known metrics: MAP (mean average precision), P@n (where n is equal to 5 and 10), AUC (area under curve).

4.1 Track 1 - Household Recommendations

The parameters used for the evaluation that have been empirically defined are the following:

- Number of users in the neighborhood $N_u = 9$;
- The first results of the rescorer used for re-ranking $N_m = 60$;
- Rescorer bosting $b = 1.25$.

[4] www.imdb.com
[5] www.wikipedia.org

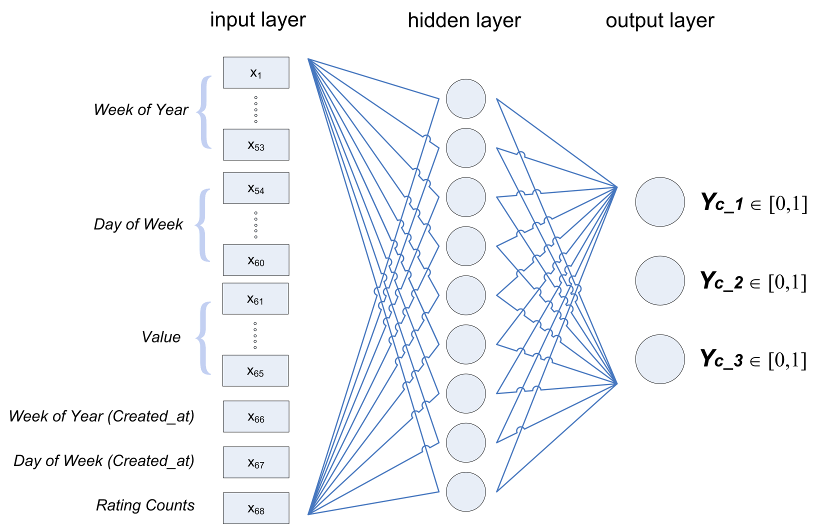

Figure 1: The architecture of the neural network.

We evaluated both the traditional CF approach and the same approach with the rescorer described in Section 3.1. Test ratings to be matched are 4482. When the household is composed of more than one person, the result lists of the algorithms of each user of the household is merged and the top ranked results are used for testing.

The results are summarized as follow::

Table 1: Track 1 evaluation results.

	CF	$CF\ w/Rescorer$
MAP	0.002	0.298
$P@5$	0.010	0.170
$P@10$	0.006	0.124

It is possible to note that the rescorer is able to increase the performance of the traditional collaborative approach, thus reaching a P@10 of 0.124 and a P@5 of 0.170.

4.2 Track 2 - Identifying Ratings

The training of the neural network was performed using a training set, which was the same size as the testing set. In order to determine the optimal values for the weights of the network, we applied a supervised learning algorithm based on gradient descent and 10-fold cross-validation to adjust the weights toward convergence. We obtained an overall high classification accuracy 71.9% (i.e., a statistical measure that assesses how well a binary classification correctly identifies or excludes objects). The values of other relevant measures are summarized in the following table:

Table 2: Neural network training error values.

0.03	Mean absolute error (MAE)
0.26	Root mean squared error (RMSE)
0,23	Relative absolute error (RAE)
44%	Root relative squared error (RRSE)

The results of the three classifiers c_1, c_2, c_3, and the classifier NN, which adopts the combined approach with the neural network, are summarized in Table 3 and Figure 2, using classification error rate by household (Average Error Rate, AER), Average Error Rate for Household ($AERH$) for each household size {2,3,4}, Household Area under the ROC Curve (AUC) and Mean Average Precision (MAP).

Table 3: Track 2 evaluation results.

	c_1	c_2	c_3	NN
MAP	0.621	0.623	0.792	0.824
AUC	0.614	0.695	0.756	0.815
AER	0.605	0.610	0.755	0.800
$AERH_2$	0.606	0.621	0.756	0.804
$AERH_3$	0.609	0.619	0.705	0.735
$AERH_4$	0.483	0.510	0.838	0.777

The results show that the approaches c_1 and c_3 provide comparable results, while the approach c_2 exhibits the worst performance. The combination of the three classifiers through a neural network provides significantly higher values of MAP and AUC than those provided by the single classifiers, thus representing the best results of our experiments. In the training set there are not households with more than four members performing ratings. Hence, we decided not to employ the measures P@5 and P@10 in this track, as their values would not be significant.

5. CONCLUSIONS

In this paper, we have discussed two context-aware movie recommendation approaches. The evaluation shows how the approach based on signal processing is able to improve the performance of traditional collaborative filtering recommendation. Machine learning techniques are able to identify users that submitted a given rating.

One of the advantages of the first approach is the chance to reuse the same recommender in different scenarios, where traditional baseline approaches have to be enhanced with contextual factors related to time.

Further work will be done along several research direc-

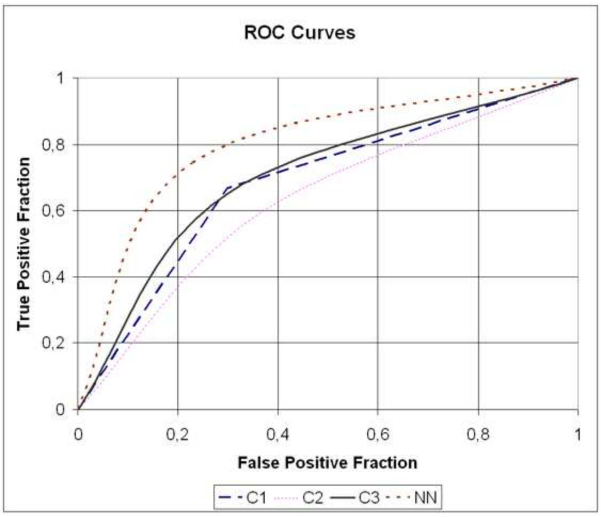

Figure 2: ROC curves for the classifiers c_1, c_2, c_3 and NN.

tions. Some factors that should be included in the recommendation process are the novelty of movies and the user authority. New movies have a higher potential of being interesting than old movies. Moreover, some collaborative approaches have tried to diversify the ratings from users, identifying more authoritative users that should be taken more into consideration when predictions have to be suggested.

6. REFERENCES

[1] L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Proceedings of the Context-aware Recommender Systems Workshop at Recsys09*, 2009.

[2] X. Bao, L. Bergman, and R. Thompson. Stacking recommendation engines with additional meta-features. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, RecSys '09, pages 109–116, New York, NY, USA, 2009. ACM.

[3] C. Biancalana, F. Gasparetti, A. Micarelli, and G. Sansonetti. An approach to social recommendation for context-aware mobile services. *ACM Trans. Intell. Syst. Technol.*, 2011. To appear.

[4] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[5] A. Brenner, B. Pradel, N. Usunier, and P. Gallinari. Predicting most rated items in weekly recommendation with temporal regression. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 24–27, New York, NY, USA, 2010. ACM.

[6] Y. Ding and X. Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM.

[7] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, New York, NY, USA, 2009. ACM.

[8] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 210–217, New York, NY, USA, 2010. ACM.

[9] T. Q. Lee, Y. Park, and Y.-T. Park. A time-based approach to effective recommender systems using implicit feedback. *Expert Syst. Appl.*, 34:3055–3062, May 2008.

[10] S. Reid and G. Grudic. Regularized linear models in stacked generalization. In *Proceedings of the 8th International Workshop on Multiple Classifier Systems*, MCS '09, pages 112–121, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[12] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating word of mouth. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM.

APPENDIX

[6] http://doi.acm.org/10.1145/2096112.2096114