Social Media Temporal Absence Recommendation: a Collaborative Filtering Perspective

Isaiah Onando Mulang'

A thesis submitted in partial fulfillment for the degree of Master of Science in Software Engineering in the Jomo Kenyatta University of Agriculture and Technology

2014

DECLARATION

I hereby declare that this thesis is my own work and effort and that it has not been submitted for a degree in any other university. Where other sources of information have been used, they have been acknowledged.

Signature: …… Date: …….

Mulang' Isaiah Onando

CERTIFICATE OF APPROVAL

This thesis has been submitted for examination with our approval as the university supervisors:

SUPERVISORS

Signature: …… Date: ……..

Professor Waweru Mwangi

JKUAT, Kenya

Signature: …… Date: …….

Dr. Joseph Orero

Strathmore University, Kenya

I dedicate this thesis to My Father Mr. Mulang' Joseph Onando (Late) and to my mother Mrs. Anastasia Achieng' Mulang'

ACKNOWLEDGEMENTS

First, I direct my thanks to the Almighty God for the blessings of life and good health throughout this thesis period and the knowledge, wisdom and strength He has provided. I will remain grateful to my supervisors especially Prof. Waweru Mwangi for his guidance since the proposal and prior until this work is complete. His knowledge and ideas were vital in shaping my research and his constant and ready availability is unparalleled. I also want to express my gratitude to the Faculty at the Department of Computing especially to Dr. Stephen Kimani for his relentless and endearing desire to see the MSc. Students through their work by organizing for presentation seminars and providing relevant information on conferences, publications and reports including general encouragement towards the completion process.

Overall, the staff and colleagues at the Department of Computing and School of Computing and Information Technology have contributed in one way or another in providing conducive environment for my research. I would like to thank Prof Takács Gabor (Széchenyi István University) for providing me with the Netflix prize data when it was no longer available online. Finally**,** I thank my family for their encouragement and motivation and being there for me when I needed emotional and social assistance.

TABLE OF CONTENTS

DECLARATION i
ACKNOWLEDGEMENTS iii
LIST OF TABLES vii
LIST OF FIGURES viii
LIST OF APPENDICES ix
LIST OF ABBREVIATIONS x
ABSTRACT xi
CHAPTER ONE 1
1.0 INTRODUCTION 1
1.1 Definition for Temporal Absence. 1
1.2 Background of Recommendations 2
1.3 Statement of the Problem 5
1.4 Research Objectives 7
1.5 Research Overview 7
1.6 Contribution 8
1.7 Thesis Organization 8
CHAPTER TWO 10
2.0 REVIEW OF LITERATURE 10
2.1 Definitions 10
2.2 General Functioning of Recommendation Algorithms 10
2.2.1 Model of Recommender Systems 12
2.3 Recommender System Strategies 13
2.2.1 Neighborhood Approaches 14
2.2.2 Improving Neighborhood CF Algorithms 16
2.3 Model – Based Techniques 18
2.3.1 Singular Value Decomposition (SVD) 19
2.3.2 Other Latent Factor Models 22
2.4 Aspect Model Approach 22
2.4 Matrix Factorization 23
2.5 The Leaning Algorithms for MF 27

2.5.1 Gradient Descent Approach 27
2.5 The SVD++ Algorithm 37
2.6 Conclusion 39
CHAPTER THREE 40
3.0 RESEARCH METHODOLOGY 40
3.1 Research Design 40
3.2 Data collection 40
3.2.1 The Netflix dataset 41
3.2.2 Social Media Data. 42
CHAPTER 4 44
ANALYSIS AND RESULTS 44
4.1 Temporal Absence Connotations in Netflix Data 44
4.1.1 Binning 44
4.1.2 Absence Connotations 45
4.1.3 Absence Modeling for E-Commerce 48
4.2 Social Media Data Analysis 49
4.2.1 Existing Recommendations 51
4.2.2 Recommendation Quality for User Centric evaluation 54
4.2.3 Absence Latent Factors 58
4.2.4 Absence Indicators 66
4.3 Conclusion 70
CHAPTER 5 72
TWITTER TEMPORAL ABSENCE MODEL 72
5.1 Tweet Factors 74
5.2 Creating the Recommendation Matrix 75
5.3 Regression Model 79
5.4 Choosing the Learning Rate 80
5.5 Prediction of Absence 82
The Absence 84
CHAPTER 6 85
CONCLUSION AND FURTHER WORK 85

6.1 Summary of the Thesis 85
6.2 Future Work 86
CHAPTER 7 87
REFERENCES 87
Appendix I: Tweet Classifier Code 91
Appendix II: Matrix Generation Code 96
Appendix III: Survey Questionnaire 107

LIST OF TABLES

Table 1: First 20 10-day bin periods……..44

Table 2: Work Place Multilayer Perceptron Network Information……..……62

Table 3: Factors affecting diminishing interest……..……...64

Table 4: Diminishing Interest Classification Accuracy…….……....66

Table 5: Descriptive statistics of the absence indicators……..…….67

Table 6: Descriptive statistics of the absence indicators……..…….68

Table 7a: Absence Indicators Correlations I……..……..…….....70

Table 8: Twitter Representation of Absence Factors…….……...72

Table 9: Twitter Temporal Absence factors matrix……...78

Table 10: Theta Gradient Descent Values for Different Alphas……..……...….80

LIST OF FIGURES

Figure 1: Model of the Recommendation Process……..…….12

Figure 2: Best – fit regression line reduces data from two dimensions into one…..……..…..20

Figure 3: Best – Second fit regression line reduces data from two dimensions into one…..…20

Figure 4: Bin ratings presence graph……..…….……..45

Figure 5: Least Bin Customer Absence graph……...……..……..46

Figure 6: Windowed comparison of absence/present to reveal finer shifts in customer numbers…….……47

Figure 7: Graph of usefulness of social media friends' suggestions……..…..…..52

Figure 8: Causes of Recommendation Irrelevance……..……..54

Figure 9: Multilayer Perceptron for Effect of Workplace Factors on Absence…….61

Figure 10: Multilayer Perceptron Classifier for Diminishing Interest……...…65

Figure 11: Absence values on Twitter activity timeline ……...…74

Figure 12: Sequence Diagram for the Matrix Builder Algorithm……..…76

Figure 13: Plot of Cost Function J for different learning rates……..81

Figure 14: Plot of Cost Function J for a learning rate 0.0166……82

LIST OF APPENDICES

Appendix I: Tweet Classifier Code……91

Appendix II: Matrix Generation Code……96

Appendix III: Survey Questionnaire……...107

LIST OF ABBREVIATIONS

CF – Collaborative Filtering.

MF – Matrix Factorization

RS – Recommender System(s)

RWR – Random walk with restart

MAE – Mean Absolute Error

kNN – k Nearest Neighbors

RMSE – Root Mean Square Error

SSE – Sum of Square Errors

RR – Ridge Regression

SVD – Singular Value Decomposition

ALS – Alternating Least Square

ACM – Association of Computing Machinery

KDD - KDD is Knowledge Discovery in Databases (as applied in KDD Cup)

ABSTRACT

The prediction accuracy of recommendations given by recommender systems and algorithms has been a major concern in recent years with a high prediction accuracy being the first goal for any recommendation. Research on evaluating recommender systems shows that algorithms in this area are still deficient in prediction accuracy but recent works prove that modeling with temporal dynamics improves the degree of recommendation accuracy. Recommendations presume the presence of users or items in the matrix. They are invariably based on similarities of users and/or items in the user-item matrix of a system, user profiles, and rating information. The major difference in operation of the recommendation algorithms is in the way the algorithms analyze data sources to develop notions of affinity between users for use in identifying well matched pairs. There is limited work focused on the temporal absence as an indicator of preference or concept drift: and hence a factor for inclusion in the recommender algorithms. This thesis has, through live data from users, determined the factors and aspects that lead to absence and indicators of absence from a social media perspective; these values are then measured and fitted into a model that predicts a user's absence duration. The deployment of this modeling is expected to indicate concept drifts, enhance affective communication and product marketing. A working temporal absence definition is realized and using Sparse Matrix manipulation with linear regression, the model is trained using gradient descent algorithm to predict future absence time and durations.

CHAPTER ONE

1.0 INTRODUCTION

1.1 Definition for Temporal Absence.

Temporal absence draws its definition from a combination of two words temporal and absence. The word temporal by definition stands for pertaining of or pertaining to time or anything enduring for a time or while only; temporary; transitory (opposed to eternal).

Absence on the other hand would mean the state of being away or the time during which a person or thing is away and by extension the fact of being without (lacking) something can also be termed as absence. In the context of online usage on which recommender systems are deployed, absence does not necessarily mean the opposite of presence for instances like a person signing in to a system then gets preoccupied by other things in their environment, hence he is physically present and his/her sign in/ sign out log indicates presence but in actual sense s/he is literally absent.

An online search of Temporal Absence gives the following statement at (zetaboards.com, 2012)

> *"Starting from sixth (or fifth?..) March up to, aprox. fifteenth, I'll be - and am now - in hospital. Might browse forum, and sometimes post, though. Perhaps will have less spare time afterwards, due to college exams and Practice."*

Temporal absence therefore, will mean partial or complete lack of activity by a user from the system or lack of activity on an item within the system for a measurable duration of time long enough to indicate probable change of behavior or interest.

1.2 Background of Recommendations

There is a broad array of algorithms based on mathematical models that have been employed on recommender systems since the 1990s. The original structuring of the recommender system models were pegged on very clear-cut correlation statistics and predictive modeling; without taking into account, the wider domain of mathematical modeling techniques in vector space analysis, linear algebra, distributions, statistics and even concepts from machine learning literature (Billsus and Pazzani, 1998) with the collaborative filtering problem being mapped to classification. On the other hand research into content based methods has been concurrently carried out and on the ways to combine the same with collaborative filtering, and to incorporate additional domain knowledge in the architecture of recommender systems.

The work by (Pine, 1993) shows that there is an increasing need for automated systems providing personalized recommendations to a user faced with an overwhelmingly large number of choices or users with divided attention due to environmental factors. For example, an increasing choice of available products is caused by companies shifting towards developing customized products that meet specific needs of different users. Further research was spurred by the public availability of datasets on the web, and the interest generated due to direct relevance to ecommerce (Melville et al., 2010).

Since Collaborative Filtering (CF) algorithms are grounded upon user profile information and usage history or customer ratings, they have been faced with two major problems namely: 1) The cold start problem brought about by data sparsity of ratings in the user – item matrix, in which they fail to provide reasonable recommendations to new users or on new (low rated) items (Schein et al., 2002) and 2) Scalability: As the numbers of users and items grow, traditional CF algorithms will suffer serious scalability problems. For example, with tens of millions of

customers $\boldsymbol{O}(\boldsymbol{M})$ and millions of items $\boldsymbol{O}(\boldsymbol{N})$ a CF algorithm with the complexity of $\boldsymbol{n}$ is already too large. As well, many systems need to react immediately to online requirements and make recommendations for all users regardless of their purchases and ratings history, which demands a higher scalability of a CF system. Resent research such as (Park S., and Chu W., 2010) have attempted to solve the cold start problem but better results are seen to be obtained from the use of Liked Open Data (LOD) which we observe from (Moreno A. et. Al, 2014, and Castillejo E. et al., 2012)

The product customization trend, coupled with e-commerce, with which customers are not provided with an option to examine the products off-the-shelf in a traditional sense, makes the problem of providing accurate personalized recommendations very important. Increasing the quality of personalized recommendations would increase customer satisfaction and loyalty, and at the same time reduce the costs caused by product return (Slobodan and Zoran, 2004). Over time, improvements to CF algorithms have allowed dimensionality reduction techniques to be brought into play to improve the quality of the recommendations as evident from research by (Sarwar et al., 2000) on using dimension reduction techniques to reduce the number of nearest neighbors and dimensions to allow CF-based algorithms to scale to large data sets and produce high-quality recommendations at the same time. On the other hand (Linden et al., 2003) addresses these scaling issues by reducing data set size using methods, such as random sampling of customers, discarding customers with few purchases, clustering similar users, and excluding very popular or unpopular items. Unfortunately, all of these methods also reduced recommendation quality (Lee et al; Springer, 2007). Matrix Factorization (Bell, Koren, & Volinsky, 2009) are techniques rooted in numerical linear algebra and statistical matrix analysis has emerged as one of the best state-of-the-art technique.

Currently, recommender systems remain an active area of research, with a dedicated ACM conference, intersecting several sub disciplines of statistics, machine learning, data mining, and information retrievals. Applications have been pursued in diverse domains ranging from recommending web pages to music, books, movies, and other consumer products (Melville et al., 2010). There are early attempts to include time based variations in usage by a customer as apparent from work by Steve Hanneke and Eric Xing who propose a family of statistical models for social network evolution over time, based on an extension of Exponential Random Graph Models. (Koren Y., 2009) adapts an approach to model the temporal dynamics along the whole time period, to allow intelligent separation of transient factors from lasting ones.

Recommendations should be able to recommend exactly what the user wants or provide a close near miss in predicting the same. When a user's requirements change, his/her preferences and activities changes and hence his behavior will change accordingly; a recommender system should be able to track such changes intuitively so as to alter recommendation to suite the new preferences, Absence from activity in the system either passively or completely can be a good indicator of shifting interest of a user. Some users are inherently dissimilar from their peers who would appear similar given their profiles or otherwise rating information. A recommendation algorithm should be able to closely monitor a user's reaction to recommendations and conclusively determine whether they are dissimilar rather than similar. For example, if a user takes a considerable amount of time before s/he reacts to a recommendation then such absence could point to an inherent dissimilarity of the user and hence a call to change operational mode of the system. Complete absence from the system can be tracked for a reasonable time, especially in social networking recommender systems, to be able to recommend to the user's friends a possible danger or mishap to their friends.

Unfortunately, current recommender systems algorithms have not yet incorporated the aspect of temporal absence as a factor. As noted by (Koren Y., 2009); a mere decay of older instances or usage of multiple separate models loses too much signal, thus degrading prediction accuracy of recommender systems. Further, modeling time based use patterns of a user and time variations of preference to items can intelligently separate transient factors from lasting ones the only twist being that even time dynamics models have still not incorporated temporal absence. If well modeled, temporal absence can point out aspects like shifting customer preferences, absence due to natural phenomena like deaths in the cases of social networks, dissimilarities of customers rather than similarity portrayed by peers and many more.

On another view point, such modeling of temporal absence can immeasurably reduce the runtime complexity of the algorithms that is if the temporal absence is done a priory hence used as a factor in dimensionality reduction (Sarwar et al., 2000 and Linden et al., 2003). Also such algorithms as proposed by (Koren Y., 2009) can easily model temporal absence and increase prediction accuracy.

1.3 Statement of the Problem

Recommendations should be able to recommend exactly what the user wants or provide a close near miss in predicting the same. When a user's requirements change, his/her preferences and activities changes and hence his behavior will change accordingly; a recommender system should be able to track such changes intuitively so as to alter recommendation to suite the new preferences, Absence from activity in the system either passively or completely can be a good indicator of shifting interest of a user. Some users are inherently dissimilar from their peers who would appear similar given their profiles or otherwise rating information, a recommendation

algorithm should be able to closely monitor a user's reaction to recommendations and conclusively determine whether they are dissimilar rather than similar; if a user takes a considerable amount of time before s/he reacts to a recommendation then such absence could point to an inherent dissimilarity of the user and hence a call to change operational mode of the system. Complete absence from the system can be tracked for a reasonable time, especially in social networking recommender systems, to be able to recommend to the user's friends a possible danger or mishap to their friends.

Unfortunately, current recommender systems algorithms have not yet incorporated the aspect of temporal absence as a factor. As Yehuda Koren (2010) notes; a mere decay of older instances or usage of multiple separate models loses too much signal, thus degrading prediction accuracy of recommender systems. Further, modeling time based use patterns of a user and time variations of preference to items can intelligently separate transient factors from lasting ones the only twist being that even time dynamics models have still not incorporated temporal absence. If well modeled, temporal absence can point out aspects like shifting customer preferences, absence due to natural phenomena like deaths in the cases of social networks, dissimilarities of customers rather than similarity portrayed by peers and many more.

Recently incorporation of LOD and social media mining has emerged as a source of user preference elicitation and recommendation datasets (Moreno A. et. al, 2014, and Castillejo E., et al, 2012). This explosion of research in social media for use in recommendation is inherently caused by the explosion of numbers and activity of social networks. It is therefore imperative that we take a look at the social media use patterns and the influence thereof on recommendation.

1.4 Research Objectives

The main objective of this research is to determine temporal absence use patterns of social networks and to and determine an approach to modeling user absence that forms an interface for temporal absence recommendations.

The following formed the specific objectives of the study

- To determine the state of the art of temporal patterns in existing recommender systems through existing literature.
- To achieve a definition of Temporal Absence in the context of Recommender Systems
- To identify factors that contribute to Temporal Absence
- To identify measures for quantifying temporal absence indicators.
- To develop a temporal absence prediction model for social network use.
- To foster application of predicted temporal absence values in social media absence recommendations

1.5 Research Overview

This thesis focuses on developing requirements and tools, which can be utilized to analyze the usage of a system in terms of time and access variations in user engagement and model a user based on their temporal usage patterns. These tools would involve:

1. Quantifying usage factors that contribute to absence.
2. Identifying absence indicators for a user.
3. Proposing a modeling technique for predicting user absence durations.
4. Employing these models in recommending user behavior to other users or apply the same in determining user's preferences.

The first three goals, would involve extensive user – centered study of existing recommender systems in which we chose Netflix prize data as our case study and assert the user temporal patterns and the existence of temporal absence.

1.6 Contribution

The thesis makes several key contributions, listed as follows:

- Defining Temporal Absence in the context of Recommender Systems: A definition derived from the existence of time based absence in usage patterns of a system and other systemic attempts in using the same, as given earlier in the background.
- Identifying factors that contribute to Temporal Absence: with concentration on Social Media, we identify the major factors that contribute to a user's absence on a system especially social media recommendation systems and provide a note on how to expand these factors in general recommendation systems. Ultimately, these become latent factors.
- Quantification of Temporal Absence indicators: As the factors are identified, we take a consideration of what shows or indicates existence of user's absence from the system; these are quantified into numerical values for modeling.
- Temporal Absence prediction model: a gradient descent trained model that predicts a user's absence duration.

1.7 Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 gives a review of literature on Recommender Systems algorithms, evolution and application domains. Chapter 3 describes the data used in this research, data collection and the sampling approaches applied. In chapter 4 a discussion on the analysis and results is provided. The results would consist of identified latent

factors and absence indicators and the correlations between these and the Temporal Absence. These factors are applied on twitter through a number of tweet characteristics and used to model temporal absence and predict an absence period as discussed in Chapter 5. Finally we conclude the thesis in chapter 6. The bibliography is in chapter 7.

CHAPTER TWO

2.0 REVIEW OF LITERATURE

There is a plethora of work related to recommender algorithms and Recommender Systems inspired by works from Artificial Intelligence, Information Retrieval and Data Mining. This section seeks to review related literature, approaches and tools previously used and elaborate the possible paths this research will endeavor to take. Starting with the definition of relevant terms, a review follows and then later on an indication of the contributions this work will introduce is provided.

2.1 Definitions.

(Gerhard F., 2011) chooses the following definition derived from (Xiao and Benbasat, 2007), as a state of the art definition for Recommender System (RS) that views the whole system as a software agent, the same definition will be adopted for this research.

"RS are software agents that elicit the interests and preferences of individual consumers, and make recommendations accordingly.

They have the potential to support and improve the quality of the decisions consumers make while searching for and selecting products online"

2.2 General Functioning of Recommendation Algorithms

A general recommendation problem can be formulated thus (Slobodan V., and Zoran O., 2004).: Assuming a database $\boldsymbol{D}$ contains $\boldsymbol{I}$ items which have been partially evaluated by $\boldsymbol{U}$ users, we are given a $\boldsymbol{U} \times \boldsymbol{I}$ matrix $\boldsymbol{M}$ with element $\boldsymbol{r_{ui}}$ representing the evaluation score of item $\boldsymbol{i}$ by user $\boldsymbol{u}$ whereby, in real world systems the matrix $\boldsymbol{M}$ is usually very sparse since users are likely to vote just for a small subset of available items. The representations $\boldsymbol{r_{u*}}, \boldsymbol{r_{*i}}$, and $\boldsymbol{r_{**}}$ denotes the average score for each user, the average score for each item, and the overall average score,

respectively. Far and large, each user will probably not vote for each item, hence $\boldsymbol{I_u}$ can be taken to denote the subset of items rated by user $\boldsymbol{u}$. Similarly, $\boldsymbol{U_i}$ denotes the subset of users that evaluated item $\boldsymbol{i}$. Given the evaluation scores of an active user $\boldsymbol{a}$ on items $\boldsymbol{I_a}$ the recommendation system task is to estimate scores of user $\boldsymbol{a}$ on the remaining items $\boldsymbol{I \setminus Ia}$.

(Slobodan V., and Zoran O., 2004), propose three simple recommendation algorithms to establish accuracy lower bounds on more complex recommendation systems: The first, MEAN, uses the overall average score $\boldsymbol{r_{**}}$ as a prediction of a preference of any user on any item. The second, GENERIC, uses the average scores $\boldsymbol{r_{*i}}$ of each item $\boldsymbol{i}$ as predictions of a new user's preferences. This algorithm is valid under the assumption that all users have similar preferences. The MEAN and GENERIC algorithms do not attempt to use information contained in scores provided by an active user.

If users have provided numerical ratings explicitly, it is likely that the subjective nature of ratings will distort overall ratings matrix $\boldsymbol{M}$. The most obvious consequence of subjective ratings would be the difference in average scores between otherwise correlated users. Therefore, the predictions of ADJUSTED_GENERIC for active user $\boldsymbol{a}$ are obtained as predictions from GENERIC adjusted by the difference $\Delta\boldsymbol{r_a}$ between the average score of user $\boldsymbol{a}$ and the average generic score over the same set of items, $\boldsymbol{I_a}$ defined as:

$$\Delta \boldsymbol{r_a} = \frac{\mathbf{1}}{|\boldsymbol{I_a}|} \sum_{\boldsymbol{i} \in \boldsymbol{I_a}} (\boldsymbol{r_{ai}} - \boldsymbol{r_{*i}}) \quad \text{(1)}$$

Where $|\boldsymbol{I_a}|$ is the cardinality of set $\boldsymbol{I_a}$. Hence the prediction of ADJUSTED_GENERIC for an active user preference on item $\boldsymbol{i}$ is calculated as:

$$\boldsymbol{p_{ai}} = \boldsymbol{r_{*i}} + \Delta \boldsymbol{r_a} \quad \text{(2)}$$

2.2.1 Model of Recommender Systems

A recommendation seeker may ask for a recommendation, or a recommender may produce recommendations with no prompting. Seekers may volunteer their own preferences, or recommenders may ask about them. Based on a set of known preferences – his/her own, the seekers, and those of other people, often people who received recommendations in the past – the recommender recommends items the seeker probably will like. In addition, the recommender may identify people with similar interests. The seeker may use the recommendation to select items from the universe or to communicate with like-minded others (Terveen L., and Hill W., 2001).

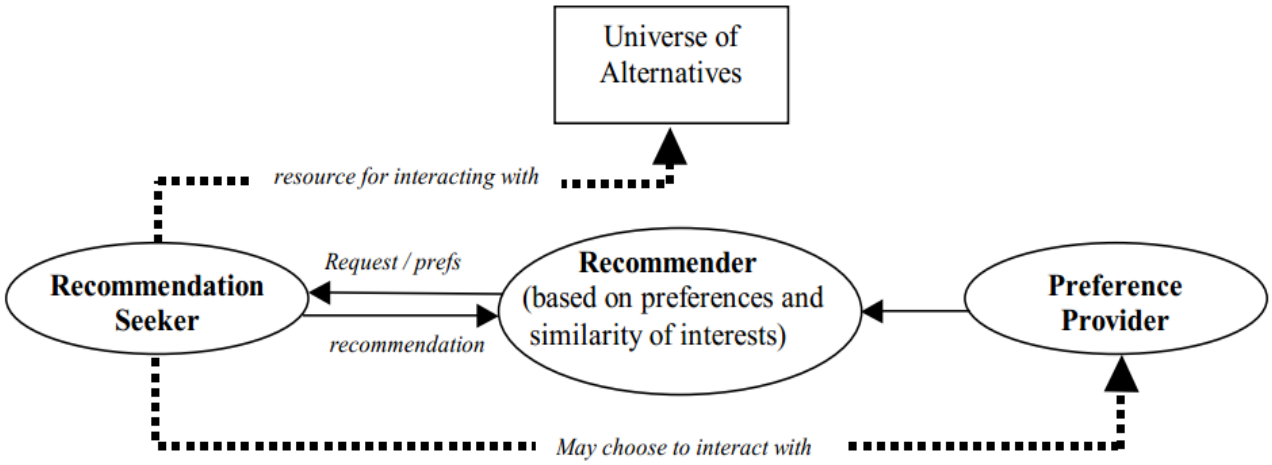


Figure 1: Model of the Recommendation Process

Recommendation is based on preferences. Thus, an automated recommender system must obtain preferences from people concerning the relevant domain (Terveen L. and Hill W., 2001). This raises a number of questions, including:

- Whose preferences are used? Those of the person seeking the recommendation, those of previous users of the system? Or perhaps preferences expressed by people in an altogether different context, such as a public forum (e.g., a chat room, bulletin board, or newsgroup)?
- How are preferences obtained? For example, do recommendation users have to express their own preferences as part of the process of seeking a recommendation?
- Are preferences expressed explicitly or implicitly?
- What incentives are there for people to offer preferences?
- What is the form of a preference? How are preferences represented?

2.3 Recommender System Strategies

In many ways Collaborative – filtering (CF) recommendation systems can be portrayed as superior to content-based alternatives, (Herlocker et al., 1999) highlight the ability to attain "serendipitous recommendations" from a system that relies on user opinions and ratings rather than content which prevents a user from having to enumerate exactly what they are looking for in a recommendation in order to have it presented to them at all. The authors conclude however that for a recommendation technology to reach the full potential, it must be combined with existing content based information filtering technology".

The underlying assumption in traditional CF models is that similar users would prefer similar items. However, CF-based models suffer from the sparsity problem and imbalance of rating data, especially for new and infrequent users. Thus, the predicted ratings from CF-models can be unreliable (Purushotham et al., 2012). There are mainly two types of CF – based approaches namely: (1) memory – based approaches or Neighborhood based Approaches, (2) model – based

approaches or the latent factor models. The memory – based approaches use either user – based approaches (Herlocker et al., 1999) or item – based approaches (Karypis, 2001) for prediction (recommendation) of ratings for items. Even though memory – based approaches are easy to implement and popular; they do not guarantee good prediction results. On the other hand, model-based approaches include several model based learning methods such as clustering models, and the latent factor models (Purushotham et al., 2012).

2.2.1 Neighborhood Approaches

The most widely cited and arguably the most commonly used CF algorithm in research is a kNN – based algorithm (Rashid et al., 2005). A subset of users is chosen based on their similarities to the active user, and a weighted combination of their ratings is used to produce the prediction of this user. Most of these approaches can be generalized by the algorithms summarized in the following steps (Melville, 2010):

1. Assign a weight to all users with respect to similarity with the active user.
2. Select $\boldsymbol{k}$ users that have the highest similarity with the active user – commonly called the neighborhood.
3. Compute a prediction from a weighted combination of the selected neighbors' ratings.

Using the preliminary descriptions in section 2.2 we acquire the user $\boldsymbol{u_i}$'s rating on item $\boldsymbol{j}$ as $(\boldsymbol{i};\boldsymbol{j}) -$ th entry of the matrix, or null, depending on whether the user $\boldsymbol{u_i}$ has rated the item $\boldsymbol{j}$, or not, respectively which can also be represented as $\boldsymbol{r_{ai}}$ $(\boldsymbol{r_{ai}} \in \boldsymbol{M})$. The user – user algorithm can be thought of working in two stages. In the first stage, similarities between every pair of users are computed and are stored as a model. Although many different formulations are possible for similarity weight calculations, the (GroupLens, 1994), proposed mechanism is the Pearson

correlation coefficient. Accordingly, the similarity weight between two users, $\boldsymbol{u_i}$ and $\boldsymbol{u_j}$ is measured by equation:

$$\boldsymbol{w_{a,u}} = \frac{\sum_{i \in I}(r_{ai} - r_{a*})(r_{ui} - r_{u*})}{\sqrt{\sum_{i \in I}(r_{ai} - r_{a*})^2 \sum_{i \in I}(r_{ui} - r_{u*})^2}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Where $\boldsymbol{I}$ is the set of items rated by both of the users i.e. $\boldsymbol{I_a \cap I_u}$. Using this similarity metric, the next step, prediction generation, is carried out as follows. Prediction on item $\boldsymbol{i}$ for the active user $\boldsymbol{a}$ is computed by picking $\boldsymbol{k}$ nearest users who have also rated item $\boldsymbol{i}$, and by applying a weighted average of deviations from the active users' means:

$$\boldsymbol{p_{a,i}} = \overline{\boldsymbol{r_a}} + \frac{\sum_{u \in K}(r_{ui} - \overline{r_a}) \times \boldsymbol{w_{a,u}}}{\sum_{u \in K} \boldsymbol{w_{a,u}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

Where $\boldsymbol{p_{a,i}}$ is the prediction for the active user $\boldsymbol{a}$ for item $\boldsymbol{i}$, $\boldsymbol{w_{a,u}}$ is the similarity between users $\boldsymbol{a}$ and $\boldsymbol{i}$, and $\boldsymbol{K}$ is the neighborhood or set of most similar users.

Alternatively, one can treat the ratings of two users as a vector in an $\boldsymbol{m}$ – dimensional space, and compute similarity based on the cosine of the angle between them, given by;

$$\boldsymbol{w_{a,u}} = \cos(\overrightarrow{\boldsymbol{r_a}}, \overrightarrow{\boldsymbol{r_u}}) = \frac{\overrightarrow{\boldsymbol{r_a}} \cdot \overrightarrow{\boldsymbol{r_u}}}{\|\overrightarrow{\boldsymbol{r_a}}\|_2 \times \|\overrightarrow{\boldsymbol{r_u}}\|_2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

When computing cosine similarity, one cannot have negative ratings, and unrated items are treated as having a rating of zero (Melville et al., 2010). There has been several other similarity measures used in the literature, including Spearman rank correlation, Kendall's τ correlation, mean squared differences, entropy, and adjusted cosine similarity (Su and Khoshgoftaar, 2009).

As noted by (Breese et al., 1998), Pearson correlation generally performs better and therefore shall be the default correlation approach in this research.

2.2.2 Improving Neighborhood CF Algorithms

(Melville et al., 2010), carries out a brief discussion on some of the current improvements to the above algorithms that have brought about improvements in performance of such algorithms we will provide a brief discussion in line with their approach below.

Item-based Collaborative Filtering

Conventional neighbourhood – based CF algorithms face scaling problems in large databases of users and items, because of the computational complexity of the search for similar users. As an alternative, (Linden et al., 2003) proposed item – item collaborative filtering in which match a user's rated items are matched to similar items as opposed to matching similar users. In practice, this approach leads to faster online systems, and results also in improved recommendations (Sarwar, et al., 2001). In this approach, similarities between pairs of items $\boldsymbol{i}$ and $\boldsymbol{j}$ are computed off – line using Pearson correlation (Melville et al., 2010), an alteration to equation 3, given by:

$$\boldsymbol{w_{i,j}} = \frac{\sum_{\boldsymbol{u}\in \boldsymbol{U_i}}(\boldsymbol{r_{ui}} - \bar{\boldsymbol{r}}_{\boldsymbol{\iota}})(\boldsymbol{r_{uj}} - \bar{\boldsymbol{r}}_{\boldsymbol{j}})}{\sqrt{\sum_{\boldsymbol{u}\in \boldsymbol{U_i}}(\boldsymbol{r_{ui}} - \bar{\boldsymbol{r}}_{\boldsymbol{\iota}})^{\,2}\ \sum_{\boldsymbol{u}\in \boldsymbol{U_i}}(\boldsymbol{r_{uj}} - \bar{\boldsymbol{r}}_{\boldsymbol{j}})^{\,2}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

From this, a simple weighted average can be used to predict the current users $\boldsymbol{a}$'s rating of item $\boldsymbol{i}$ (Melville et al., 2010) as follows:

$$\boldsymbol{p_{a,i}} = \frac{\sum_{\boldsymbol{j}\in \boldsymbol{K}}\ \boldsymbol{r_{aj}}\ \boldsymbol{w_{i,j}}}{\sum_{\boldsymbol{u}\in \boldsymbol{K}}\ \boldsymbol{w_{a,u}}} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

For item-based collaborative filtering too, one may use alternative similarity metrics such as adjusted cosine similarity.

Significance Weighting

It is common for the active user to have highly correlated neighbours that are based on very few co-rated (overlapping) items. These neighbours based on a small number of overlapping items tend to be bad predictors. One approach to tackle this problem is to multiply the similarity weight by significance weighting factor, which devalues the correlations based on few co-rated items (Herlocker et al., 1999).

Default Voting

This is an alternative approach to dealing with correlations based on very few co-rated items that assumes a default value for the rating for items that have not been explicitly rated. In this way one can now compute correlation (Eq. 3) using the union of items rated by users being matched as opposed to the intersection as shown by (Breese et al.,1998) to improve collaborative filtering.

Inverse User Frequency

Another improvement to CF algorithms forwarded by Breese et al (1998) that tries to account for the fact that universally liked items or items that have been rated by all users are not as useful as less common items and is computed as $\boldsymbol{f_i = \log \frac{n}{n_i}}$ where $\boldsymbol{n_i}$ is the number of users who have rated item $\boldsymbol{i}$ out of the total number of $\boldsymbol{n}$ users. Based on the assumption that the items which are universally loved or hated are rated more frequently than others, the Inverse User Frequency is applied to similarity based CF by multiplying the original rating of item $\boldsymbol{i}$ with the factor $\boldsymbol{f_i}$.

Case Amplification

Introduced by (Breese et al., 1998) and is a transformation of the original weights in Eq. (4) to

$$\boldsymbol{w'_{a,u} = w_{a,u} \cdot \left|w_{a,u}\right|^{\rho-1}} \quad \text{.....} \quad (8)$$

This provides an avenue to favour users with high similarity to the active user (Melville et al., 2010)

Neighborhood models are most effective at detecting much localized relationships. They rely on a few significant neighborhood-relations, often ignoring the vast majority of ratings by a user. Consequently, these methods are unable to capture the totality of weak signals encompassed in all of a user's ratings (Koren Y., 2008).

2.3 Model – Based Techniques

Model-based techniques are such approaches that provide recommendations by estimating parameters of statistical models for user ratings. For example, (Billsus and Pazzani, 2009) describe an early approach to map CF to a classification problem, and build a classifier for each active user representing items as features over users and available ratings as labels, possibly in conjunction with dimensionality reduction techniques to overcome data sparsity issues. Other predictive modelling techniques have also been applied in closely related ways.

There are majorly two modeling techniques used in Recommender algorithms, namely Aspect modeling and Matrix Factorization (MF), this section will explain the two techniques and then concentrate on MF as the modeling of choice for this research with a clear discussion for the motivation behind the same. As noted by (Koren Y., 2008), Latent factor models, such as Singular Value Decomposition (SVD), comprise an alternative approach to CF from the neighborhood based approaches by transforming both items and users to the same latent factor space, thus making them directly comparable. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback.

Latent factor and MF models have recently emerged as state-of-the-art methodology in this class of techniques (Bell et al., 2009). Unlike neighbourhood based methods that generate recommendations based on statistical notions of similarity between users, or between items, latent factor models assume that the similarity between users and items is simultaneously induced by some hidden lower – dimensional structure in the data (Melville, 2010).

2.3.1 Singular Value Decomposition (SVD)

Kirk Baker (March, 2005)

Singular value decomposition (SVD) a mathematical technique from linear algebra that is most often deployed in model based recommendations. This section will give a brief description of the functioning of this technique. SVD can be looked at from three mutually compatible points of view. On the one hand, we can see it as a method for transforming correlated variables into a set of uncorrelated ones that better expose the various relationships among the original data items. At the same time, SVD is a method for identifying and ordering the dimensions along which data points exhibit the most variation. This ties in to the third way of viewing SVD, which is that once we have identified where the most variation is, it's possible to find the best approximation of the original data points using fewer dimensions. Hence, SVD can be seen as a method for data reduction.

As an illustration of these ideas, consider the 2 – dimensional data points in Figure 2. The regression line running through them shows the best approximation of the original data with a 1-dimensional object (a line). It is the best approximation in the sense that it is the line that minimizes the distance between each original point and the line. If we drew a

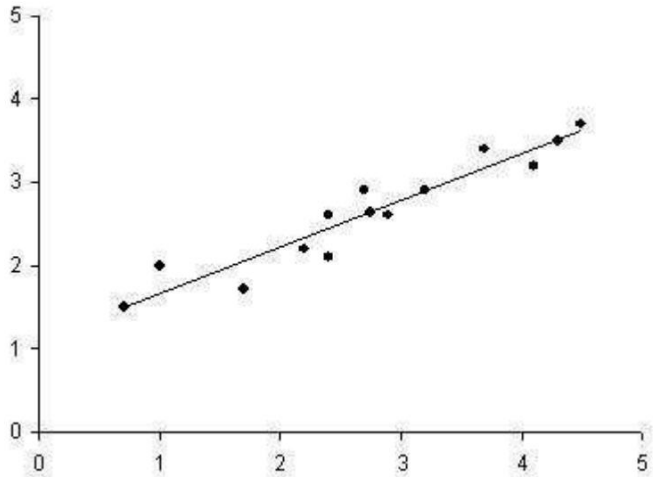


Figure 2 Best – fit regression line reduces data from two dimensions into one.

Perpendicular line from each point to the regression line, and took the intersection of those lines as the approximation of the original data point, we would have a reduced representation of the original data that captures as much of the original variation as possible. There is a second regression line, perpendicular to the first, shown in Figure 3 which captures as much of the variation as possible along the second dimension of the original dataset. It does a poorer job of approximating the original data because it corresponds to a dimension exhibiting less variation to begin with. It is possible to use these regression lines to generate a set of uncorrelated data points that will show sub groupings in the original data not necessarily visible at first glance.

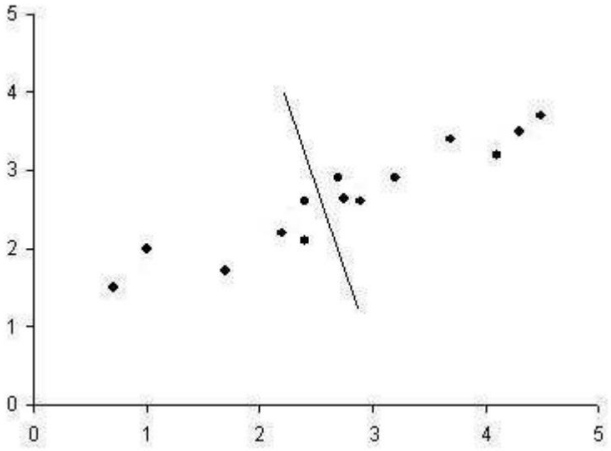


Figure 3: Best – Second fit regression line reduces data from two dimensions into one.

These are the basic ideas behind SVD: taking a high dimensional, highly variable set of data points and reducing it to a lower dimensional space that exposes the substructure of the original data more clearly and orders it from most variation to the least. What makes SVD practical for Non Linear Programming applications is that you can simply ignore variation below a particular threshold to massively reduce your data but be assured that the main relationships of interest have been preserved.

SVD is based on a theorem from linear algebra which says that a rectangular matrix $\boldsymbol{A}$ can be broken down into the product of three matrices: an orthogonal matrix$\boldsymbol{U}$, a diagonal matrix $\boldsymbol{S}$, and the transpose of an orthogonal matrix $\boldsymbol{V}$. The theorem is usually presented in this form:

$$\boldsymbol{A_{mn} = U_{mm}\ S_{mn} V^T_{nn}} \quad \text{.....} \quad (9)$$

Where $\boldsymbol{U^T U = I\ ,\ V^T V = I}$ the columns of $\boldsymbol{U}$ are orthonormal eigenvectors of $\boldsymbol{AA^T}$, the columns of $\boldsymbol{V}$ are orthonormal eigenvectors of $\boldsymbol{AA^T}$, and $\boldsymbol{S}$ is a diagonal matrix containing the square roots of eigenvalues from $\boldsymbol{U}$ or $\boldsymbol{V}$ in descending order.

Reduced singular value decomposition is the mathematical technique underlying a type of document retrieval and word similarity method variously called Latent Semantic Indexing or Latent Semantic Analysis. The insight underlying the use of SVD for these tasks is that it takes the original data, usually consisting of some variant of a word document matrix, and breaks it down into linearly independent components. These components are in some sense an abstraction away from the noisy correlations found in the original data to sets of values that best approximate the underlying structure of the data set along each dimension independently.

Because the majority of those components are very small, they can be ignored, resulting in an approximation of the data that contains substantially fewer dimensions than the original. SVD

has the added benefit that in the process of dimensionality reduction, the representation of items that share substructure become more similar to each other, and items that were dissimilar to begin with may become more dissimilar as well.

Latent factor models are generally effective at estimating overall structure that relates simultaneously to most or all items. However, these models are poor at detecting strong associations among a small set of closely related items, precisely where neighborhood models do best.

2.3.2 Other Latent Factor Models

Matrix factorization approaches are considered as latent factor s models, because each user and item gets associated with factors that are discovered by the learning algorithm. Latent factor models have been extensively studied in the field of CF: probabilistic latent semantic analysis restricted Boltzmann machines, maximum margin MF. For reference on these see (Istv´an Pil´aszy, 2009).

2.4 Aspect Model Approach

The Aspect Model is a probabilistic latent space model, which models user interests as a mixture of preference. The latent class variables: $\boldsymbol{f} \in \boldsymbol{F} \coloneqq \{\boldsymbol{f_1}, \boldsymbol{f_2}, \dots, \boldsymbol{f_k}\}$ are associated with each user u and each item $\boldsymbol{i}$. Users and items are independent from each other given the latent class variable $\boldsymbol{f}$. The probability for each observation tuple $(\boldsymbol{u}, \boldsymbol{i}, \boldsymbol{r})$ is calculated as follows:

$$\boldsymbol{p}(\boldsymbol{r}|\ \boldsymbol{i}, \boldsymbol{u}) = \sum_{\boldsymbol{f} \in \boldsymbol{F}} \boldsymbol{p}(\boldsymbol{r}|\boldsymbol{f}, \boldsymbol{i})\ \boldsymbol{p}(\boldsymbol{f}|\boldsymbol{u}) \quad \dots\dots \quad (10)$$

Where $\boldsymbol{p}(\boldsymbol{f}|\boldsymbol{u})$ is a multinomial distribution and stands for the likelihood for user $\boldsymbol{u}$ to be in the latent class$\boldsymbol{f}$. is the likelihood of assigning item $\boldsymbol{i}$ with rating r for class $\boldsymbol{f}$. In order to achieve better performance, the training ratings of each user are normalized with zero mean and variance

1 (Hofmann, 2003). The parameter $\boldsymbol{p(r|f,i)}$ is a Gaussian distribution $\boldsymbol{N(\mu_{i,f}, \sigma_{i,f})}$ with latent class mean $\boldsymbol{\mu_{i,f}}$ and standard deviation $\boldsymbol{\sigma_{i,f}}$.

2.4 Matrix Factorization

A detailed discussion of these models is provided by (Karatzoglou A., and Weimer M., 2010), this section gives an overview of the MF modeling approach. As noted by (Melville, 2010), Matrix Factorization techniques are a class of widely successful latent factor models where users and items are simultaneously rep-resented as unknown feature vectors (column vectors). Inner products of these vectors are used to approximate the known preference rationings with respect to some loss measure.

Matrix Factorization (MF) is currently the most effective and efficient recommender algorithm. It is similar to methods like Principal Component Analysis and Singular Value Decomposition in that it tries to extract patterns embedded in data. This is done by decomposing one matrix into two matrices, such that the inner product of these matrices reproduces the original matrix (Mark Graus, 2011). The intuition behind using MF to solve this problem is that there should be some latent features that determine how a user rates an item. For example, two users would give high ratings to a certain movie if they both like the actors/actresses of the movie, or if the movie is an action movie, which is a genre preferred by both users. Hence, if we can discover these latent features, we should be able to predict a rating with respect to a certain user and a certain item, because the features associated with the user should match with the features associated with the item.

In trying to discover the different features, we also make the assumption that the number of features would be smaller than the number of users and the number of items. It should not be

difficult to understand this assumption because clearly it would not be reasonable to assume that each user is associated with a unique feature (although this is not impossible). And anyway if this is the case there would be no point in making recommendations, because each of these users would not be interested in the items rated by other users. Similarly, the same argument applies to the items (URL-2, 2012).

In Matrix Factorization methods for Collaborative Filtering, the known data is interpreted as a sparse matrix $\boldsymbol{M} \in \mathbb{R}^{\boldsymbol{n} \times \boldsymbol{m}}$. In this case of a recommender algorithm, it tries to describe the user – item matrix $\boldsymbol{M}$ as a $\boldsymbol{n} \times \boldsymbol{m}$ matrix (with $\boldsymbol{n}$ the number of users and $\boldsymbol{m}$ the number of items in the catalogue (Mark Graus, 2011). The cells in $\boldsymbol{M}$ denoted as $\boldsymbol{M_{i,j}}$ contain the rating of item $\boldsymbol{j}$ by the corresponding user $\boldsymbol{i}$, if such a rating is known. The predicted rating $\boldsymbol{F_{i,j}}$ of item $\boldsymbol{j}$ by user $\boldsymbol{i}$ is modeled as a linear combination of item factors $\boldsymbol{Q_{j*}} \in \mathbb{R}^{\boldsymbol{k}}$ and user factors $\boldsymbol{U_{i*}} \in \mathbb{R}^{\boldsymbol{k}}$ **,** given the assumption that we would like to discover K latent features:

$$\boldsymbol{F_{i,j}} = \boldsymbol{U_{i*}}, \boldsymbol{Q_{*j}} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

Where $\boldsymbol{U_{i*}}$ is the factor vector for user $\boldsymbol{i}$ and $\boldsymbol{Q_{*j}}$ the factor vector for item $\boldsymbol{j}$. Let $\boldsymbol{U} \in \mathbb{R}^{\boldsymbol{n} \times \boldsymbol{k}}$ denote the matrix of all user factor vectors and $\boldsymbol{Q} \in \mathbb{R}^{\boldsymbol{m} \times \boldsymbol{k}}$ the matrix of all item factor vectors. We can then express this prediction rule as a matrix product:

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{Q}' \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

That is, each row of $\boldsymbol{U}$ would represent the strength of the associations between a user and the features. Similarly, each row of $\boldsymbol{Q}$ would represent the strength of the associations between an item and the features. As expressed by (Istv´an Pil´aszy, 2009), if we consider the matrices as linear transformations, the approximation can be interpreted as follows: matrix $\boldsymbol{Q}$ is a transformation from $\mathbb{R}^{\boldsymbol{M}}$ into $\mathbb{R}^{\boldsymbol{K}}$, and matrix $\boldsymbol{U}$ is a transformation from $\mathbb{R}^{\boldsymbol{K}}$ into $\mathbb{R}^{\boldsymbol{N}}$, and $\boldsymbol{M}$ is a

transformation from $\mathbb{R}^{\boldsymbol{M}}$ into $\mathbb{R}^{\boldsymbol{K}}$. Typically, K≪N and K≪M, therefore the intermediate K-dimensional vector space acts as a bottleneck in the approximation of $\boldsymbol{M}$ as a sequence of two linear transformations.

That is to say that the number of parameters to describe $\boldsymbol{M}$ can be reduced from $|\boldsymbol{M}|$ to NK + MK, if K is small. However, there are cases when NK + MK is larger than $|\boldsymbol{M}|$, for example, when K is greater than the average number of ratings per user.

A good learning algorithm should be able to handle these cases as well, since even for relatively small values of K, there will be many users who have less than K ratings. Note that $\boldsymbol{U}$ and $\boldsymbol{Q}$ typically contain real numbers, even when $\boldsymbol{M}$ contains only integers (Istv´an Pil´aszy, 2009).

To get the prediction of a rating of an item $\boldsymbol{q_j}$ by user $\boldsymbol{u_i}$ we can calculate the dot product of the two vectors corresponding to $\boldsymbol{u_i}$ and $\boldsymbol{q_j}$ (URL-2, 2012):

$$\hat{\boldsymbol{r}}_{\boldsymbol{ij}} = \boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}\, \boldsymbol{q_j} \;=\; \sum_{\boldsymbol{k=1}}^{\boldsymbol{k}} \boldsymbol{u_{ik}}\, \boldsymbol{q_{kj}} \quad \text{..... (13)}$$

Where $\boldsymbol{u_i}$ is the transpose of the $\boldsymbol{i}^{\text{th}}$ row of $\boldsymbol{U}$, the user feature vector; $\boldsymbol{q_j}$ is the transpose of the $\boldsymbol{j}^{\text{th}}$ row of $\boldsymbol{Q}$, the user feature vector $\boldsymbol{u_{ik}}$ is the (i,k)-th element of $\boldsymbol{U}$, and the item feature vector $\boldsymbol{q_{kj}}$ is the (u,k)-th element of $\boldsymbol{Q}$.

The major challenge is computing the mapping of each item and user to factor vectors $\boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}\, \boldsymbol{q_j} \in \mathbb{R}^{\boldsymbol{d}}$**.** After the recommender system completes this mapping, it can easily estimate the rating a user will give to any item by using equation 13. Such a model is closely related to SVD, which is a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD in the collaborative filtering domain requires factoring the user-item rating matrix. This often raises difficulties due to the high portion of missing values

caused by sparseness in the user-item ratings matrix. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting (Koren Y., 2009).

Earlier systems relied on imputation to fill in missing ratings and make the rating matrix dense.

However, imputation can be very expensive as it significantly increases the amount of data. In addition, inaccurate imputation might distort the data considerably. When learning a MF model, the aim is to estimate $\boldsymbol{U}$ and $\boldsymbol{I}$ in such a way that the model predictions $\boldsymbol{F}$ minimize a loss $\boldsymbol{L}$ on the training set $\boldsymbol{M}$:

$$U, I := argmin_{U,I}\, L(F = UI', M) \quad (15)$$

However, optimizing this will typically yield poor predictive performance due to overfitting. Thus, a regularization function $\boldsymbol{\Omega(F)}$ is added for capacity control and thus overfitting prevention. This leaves us with the following objective function (Karatzoglou A., and Weimer M., 2010):

$$U, I := argmin_{U,M}\, L(F = UI', M) + \lambda\Omega(F) \quad (16)$$

Here, $\boldsymbol{\lambda}$ is a constant that is used to control the trade – off between the regularization and the performance of the system on the known training data and is usually determined by cross – validation. Typically, the regularizer $\boldsymbol{\Omega(F)}$ is chosen to be the sum of the Frobenius (L2) norms of the matrices $\boldsymbol{U}$ and $\mathbf{I}$. The subsequent discussion is the methods deployed in finding the matrices $\boldsymbol{U}$ and $\boldsymbol{I}$. The model suggested by (Paterek A., 2007 and Koren Y., 2008, Jahrer M. et al. 2010), models directly the observed ratings only, while avoiding overfitting through a regularized model. The system minimizes the regularized squared error on the set of known ratings, which is an implementation of the above general form:

$$\min_{u^*, q^*} \sum_{i,j \in k} (r_{ij} - \hat{r}_{ij})^2 + \lambda(\|u_i\|^2 + \|q_j\|^2) \quad \text{.....} \quad (17)$$

Where $\hat{r}_{ij}$ is obtained from equation 13.

2.5 The Leaning Algorithms for MF

Two approaches to minimizing Equation 17 are: gradient descent and alternating least squares (ALS) as discussed in the next section.

2.5.1 Gradient Descent Approach

One among the numerous techniques used to approach this problem of finding the two matrices U and I is to first initialize the two matrices with some values, calculate how `different' their product is to M, and then try to minimize this difference iteratively. Such a method is called gradient descent, aiming at finding a local minimum of the difference (URL-2, 2012).

2.5.1.1 Incremental Gradient Descent

There are many variations of incremental gradient descent methods that exist. Basically, these methods iterate through the available ratings, and changes model parameters such that the actual prediction error is decreased. Here is the pseudo-code of one variant (Istv´an Pil´aszy, 2009):

```
1. foreach epoch e
2.     foreach user u
3.         foreach item j rated by u
4.             Calc err = M − U Q^T
5.             foreach feature index k
6.                 update u_ik and q_jk to decrease err_ij
7.             endfor
8.         endfor
9.     endfor
10. endfor
```

Algorithm 1: Pseudo-code for gradient descent.

The focus is on the large number of possible alternatives, which are more or less useful. Algorithm 1 iterates through all users, and then queries all items rated by user u. It calculates the error of the prediction of the model (only the error on the $(\boldsymbol{i},\boldsymbol{j})$ −th rating matters), and then updates all model parameters of the active user and item, namely the $\boldsymbol{u_{ik}}$ and $\boldsymbol{q_{jk}}$ variables for each$\boldsymbol{k}$, to better predict the actual rating. The iteration through all the ratings is repeated many times (the number of epochs), until the prediction error on a given test stops decreasing.

There are a number of decision points (Istv´an Pil´aszy, 2009):

- Should we iterate over all items, or just items rated by the user? This is very straight forward in this algorithm: use only existing ratings. However, some previous CF algorithm applied the well-known SVD method on the whole R. In that case, the unknown ratings were replaced by a baseline prediction.
- The order of iteration: should we order ratings by $(\boldsymbol{i},\boldsymbol{j})$ or by $(\boldsymbol{j},\boldsymbol{i})$? It has been observed that amongst many possible orderings, iterating over users in an arbitrary order, and for each user, taking the ratings in an increasing chronological order, that is, starting from the oldest and ending with the newest yields the best RMSE. In the followings, I refer to this ordering as the $(\boldsymbol{u},\boldsymbol{d_{ij}})-$ ordering, although the order of users are arbitrary, i.e. they need not to be ordered exactly according to their numeric index , any permutation is sufficient. The $(\boldsymbol{u},\boldsymbol{d_{ij}})-$ ordering is even better than ordering only by$(\boldsymbol{d_{ij}})$.
- The point of error calculation: In algorithm 1, error is calculated before updating the two K-dimensional vector. However, there may be other variants, for example: error is calculated before updating $\boldsymbol{u_{ik}}$ or $\boldsymbol{q_{\mathrm{ik}}}$ for a specific $\boldsymbol{k}$, or errors are calculated before iterating through all ratings of $\boldsymbol{u}$ (updating in a batch mode).

- Updating variables: Even the variable update process can be made more complicated than above: we can collect the changes going to be made to $\boldsymbol{u_{ik}}$ and $\boldsymbol{q_{ik}}$, and then update them at some point when the changes are summarized .

(Koren Y., 2009) adopts a stochastic gradient descent optimization of equation 17 initially popularized by (Simon Funk, URL-3) wherein the algorithm loops through all ratings in the training set. For each given training case, the system predicts $\boldsymbol{r_{ij}}$ and computes the associated prediction error:

$$e_{ij} \stackrel{\text{def}}{=} r_{ij} - q_j^T u_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (18)$$

Then it modifies the parameters by a magnitude proportional to $\boldsymbol{\gamma}$ in the opposite direction of the gradient, yielding:

$$q_j \leftarrow q_j + \gamma.\left(e_{ij}\,.\,u_i - \lambda\,.\,q_j\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (19a)$$

$$u_i \leftarrow u_i + \gamma.\left(e_{ij}\,.\,q_j - \lambda\,.\,u_i\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (19b)$$

This popular approach (Paterek A., 2007 and Simon Funk, URL-3) combines implementation ease with a relatively fast running time. Yet, in some cases, it is beneficial to use ALS optimization (Koren Y., 2009).

Biased Regularized Incremental Simultaneous Matrix Factorization.

According to (Istv´an Pil´aszy, 2009), a regularized incremental simultaneous MF can be modeled as follows: Considering that the matrix $\boldsymbol{M}$contains many unknown ratings which cannot be represented by zero, the approximation of $\boldsymbol{M}$ can be defined as follows:

$$\hat{\boldsymbol{r}}_{ij} = \sum_{k=1}^{K} \boldsymbol{u_{ik}} \boldsymbol{q_{jk}} = \boldsymbol{u_i^T q_j} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (20)$$

$$\boldsymbol{e_{ij}} = \boldsymbol{r_{ij}} - \hat{\boldsymbol{r}}_{ij} \text{ for} (\boldsymbol{i}, \boldsymbol{j}) \in \boldsymbol{R} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (21)$$

$$\boldsymbol{l_{ij}} = \frac{\boldsymbol{1}}{\boldsymbol{2}} \boldsymbol{e_{ij}^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (21a)$$

$$SSE = \sum_{(i,j \in R)} \boldsymbol{e_{ij}^2} = \sum_{(i,j \in R)} \left(\boldsymbol{r_{ij}} - \sum_{k=1}^{K} \boldsymbol{u_{ik} q_{jk}}\right)^2 \quad \text{(21b)}$$

$$L = \frac{1}{2} SSE = \sum_{(i,j) \in R} \boldsymbol{l_{ij}} \quad \text{(21c)}$$

$$\text{RMSE} = \sqrt{\frac{SSE}{|R|}} \quad \text{(21d)}$$

$$(\boldsymbol{U}*, \boldsymbol{Q}*) = argmin\, SSE(\boldsymbol{P}, \boldsymbol{Q}) = argmin\, L(\boldsymbol{P}, \boldsymbol{Q}) = argmin\, RMSE(\boldsymbol{P}, \boldsymbol{Q}) \quad \text{(21}e\text{)}$$

In equation $21a$, $\boldsymbol{l_{ij}}$ denotes the training loss associated with the (i, j) −th rating, which will be important later on. $\boldsymbol{L}$ is the training loss, which is the sum of all training losses measured on each training example. Eq. $(21a)$ states that the optimal $\boldsymbol{P}$ and $\boldsymbol{Q}$ minimize the sum of squared training errors.

Minimizing RMSE can be seen as a weighted low -rank approximation of R. Weighted

Low – rank approximations try to minimize the objective function $SSE_W = \sum_{i=1}^{N} \sum_{j=1}^{M} \boldsymbol{w_{ij}} . \boldsymbol{e_{ij}^2}$; where $\boldsymbol{w_{ij}}$ - s are predefined non – negative weights. For collaborative filtering problems, $\boldsymbol{w_{ij}}$ is 1 for known ratings, and 0 for unknown ratings. When the rank of $((\boldsymbol{w_{ij}}))$ is 1, all local minima are global minima. However, when it is greater than 1, as in the case of collaborative filtering, it can be shown by counter examples (Srebro and Jaakkola, 2003) that this statement does not hold any more meaning that gradient method can find only local minima. Nonetheless, different random initializations of $\boldsymbol{U}$ and $\boldsymbol{Q}$ lead to almost the same result, thus, sticking in a local minima is usually not a problem as shown by (Takács et al., 2007).

For the incremental gradient descent method, suppose we are at the $(\boldsymbol{i}, \boldsymbol{j})$ −th training example, and $\boldsymbol{r_{ij}}$ and its approximation $\hat{\boldsymbol{r}}_{\boldsymbol{ij}}$, are given; we compute the gradient of $\boldsymbol{l_{i,j}}$ thus:

$$\frac{\partial}{\partial u_i} l_{ij} = -e_{ij} . q_j, \frac{\partial}{\partial q_j} l_{ij} = -e_{ij} . u_i \quad \text{(22)}$$

Then we update the weights in the direction opposite to the gradient i.e. we change the weights of P and Q to decrease the square of actual error thus better approximating $\boldsymbol{r_{ij}}$ as adopted by (Koren Y., 2009), previously shown:

$$u_i' = u_i + \eta \,.\, e_{ij} \,.\, q_j, \quad \ldots\ldots\ldots\ldots \quad (22a)$$

$$q_j' = q_j + \eta \,.\, e_{ij} \,.\, p_i, \quad \ldots\ldots\ldots\ldots \quad (22b)$$

Here, $\boldsymbol{\eta}$ is the learning rate. When training is finished, each value of $\boldsymbol{M}$ can be computed easily using$(\boldsymbol{i})$, even at unknown positions. In other words, the model $(\boldsymbol{U}*, \boldsymbol{Q}*)$ provides a description of how an arbitrary user would rate any item (Istv´an Pil´aszy, 2009).

Adding Regularization

As earlier noted, overfitting is a possible occurrence in this process especially in the cases where the user has fewer rating than $\boldsymbol{K}$, assuming that the feature vectors of the items rated by the user are linearly independent and $\boldsymbol{Q}$ does not change; there exists a user feature vector with zero training error. Thus, there is a potential for overfitting, if $\boldsymbol{\eta}$ and the extent of the change in $\boldsymbol{Q}$ are both small (Istv´an Pil´aszy, 2009). We avoid such overfitting by penalizing the square of the Euclidean norm of weights; a common technique applied in Machine learning methods like the Support Vector Machines (SVM) and ridge regression. Penalizing the weights result in a new optimization problem which modifies eq.$(21a)$:

$$l_{ij} = \frac{1}{2}(e_{ij}^2 + \lambda \,.\, u_i^T u_i + \lambda \,.\, q_j^T q_j) \quad \ldots\ldots\ldots\ldots \quad (23)$$

$$L = \sum_{(i,j)\in R} l_{ij} = \frac{1}{2}\sum_{(i,j)\in R}(e_{ij}^2 + \lambda \,.\, u_i^T u_i + \lambda \,.\, q_j^T q_j) \ldots\ldots\ldots\ldots \quad (24)$$

$$(\boldsymbol{U}*, \boldsymbol{Q}*) = argmin\, L(\boldsymbol{P}, \boldsymbol{Q}) \quad \ldots\ldots\ldots\ldots \quad (25)$$

Here $\boldsymbol{\lambda} \geq 0$ is the regularization factor. Note that minimizing L is no longer equivalent to minimizing SSE, unless $\boldsymbol{\lambda} = 0$, in which case we get back the basic method. Note that in the above formula of $\boldsymbol{L}$, the penalization of $\boldsymbol{u_i}$ is proportional to n_i, the number of items rated by user $\boldsymbol{i}$. Similarly, the penalization of $\boldsymbol{q_j}$ is proportional to n_j, the number of ratings on item $\boldsymbol{j}$.

With this regularization approach, the gradient of $\boldsymbol{l_{ij}}$ is the following:

$$\frac{\partial}{\partial u_i} l_{ij} = -e_{ij} \,.\, q_j + \lambda \,.\, u_i \,,\; \frac{\partial}{\partial q_j} l_{ij} = -e_{ij} \,.\, u_i + \lambda \,.\, q_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (26)$$

The weights are updated in a similar fashion as shown previously in equations 22a and 22b:

$$u_i' = u_i + \eta \,(e_{ij} \,.\, q_j + \lambda \,.\, u_i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (26a)$$

$$q_j' = q_j + \eta \left(e_{ij} \,.\, p_i + \lambda \,.\, q_j\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (26b)$$

Besides penalizing the magnitude of weights, an Incremental Simultaneous Matrix Factorization (ISMF) also applies early stopping: when the prediction error on a validation set stops decreasing, it stops the learning procedure. Thus, $\boldsymbol{U}^*$ and $\boldsymbol{Q}^*$ differs from eq.($\boldsymbol{v}$), because of the early stopping. Note that the matrices are initialized randomly. If both P and Q are initialized with a constant value, that is, both are rank 1, the weight update will not increase the rank, which is equivalent to the K = 1 case. Random initialization is a simple way to avoid this, allowing matrices to be full – rank.

Adding Biases

Naturally some users tend to rate all items higher or lower than the average. The same may hold for items: some items can be very popular (Istv´an Pil´aszy, 2009). Although MF can reconstruct the original matrix exactly when K is large enough and $\boldsymbol{\lambda} = 0$, this is not the case when overfitting is to be avoided. There is a straightforward way to extend the above model to be able

to directly model this phenomenon, by extending MF with biases for users and items. The bias feature idea in the context of matrix factorization was mentioned first by (Paterek A., 2007). Many other authors have also used some method to account for biases. For example, (Sarwar et al., 2000), subtracted user average before training. Simon Funk applied double centering, first by subtracting the movie average, then the user average of the residuals. The advantage of Bias based modeling and Paterek's approach is that these parameters are trained with other model parameters simultaneously.

This can be considered in two different but mathematically equivalent ways. The first way fixes the first column of $\boldsymbol{P}$ and the second column of $\boldsymbol{Q}$ to the constant value of 1, which is, they are initialized to 1 and not updated. The second way is to introduce two new vectors, $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{N}\times\boldsymbol{1}}$ and $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{M}\times\boldsymbol{1}}$, and modify the formula of predictions:

$$\boldsymbol{r_{ij}} = \boldsymbol{b_i} + \boldsymbol{c_j} + \boldsymbol{u_i^T q_j} \quad \ldots\ldots \quad (27)$$

Where $\boldsymbol{c_j}$ is the item bias (related to the average rating of item), and $\boldsymbol{b_i}$ is the user bias by which a user offsets his ratings for all movies. This simple extension speeds up the training phase and yields a more accurate model with better generalization performance. This bias based improvement is always superior to just the regularized form in terms of both the accuracy and the running time, hence worth being a recommended basic MF method (Istv´an Pil´aszy, 2009). The task of finding an appropriate $\boldsymbol{P}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ can be defined as an optimization problem:

$$l_{ij} = \frac{1}{2}(e_{ij}^2 + \lambda\,.\,u_i^T u_i + \lambda\,.\,q_j^T q_j + \lambda\,.\,b_i^2 + \lambda\,.\,c_j^2) \quad \ldots\ldots \quad (28)$$

$$L = \sum_{(i,j)\in R} l_{ij} \quad \ldots\ldots \quad (29)$$

$$(\boldsymbol{U}*, \boldsymbol{Q}*, \boldsymbol{b}^*, \boldsymbol{c}^*) = argmin\, L(\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{b}, \boldsymbol{c}) \quad \ldots\ldots \quad (30)$$

The gradient is the following:

$$\frac{\partial}{\partial u_i} l_{ij} = -e_{ij} \,.\, q_j + \lambda u_i \quad \text{..... (31a)}$$

$$\frac{\partial}{\partial q_j} l_{ij} = -e_{ij} \,.\, u_i + \lambda q_j \quad \text{..... (31b)}$$

$$\frac{\partial}{\partial b_i} l_{ij} = -e_{ij} + \lambda b_i \quad \text{..... (31c)}$$

$$\frac{\partial}{\partial c_j} l_{ij} = -e_{ij} + \lambda c_j \quad \text{..... (31d)}$$

The weights are then calculated in the direction opposite to the gradient as shown below:

$$u_i' = u_i + \eta \,(e_{ij} \,.\, q_j - \lambda u_i) \quad \text{..... (31e)}$$

$$q_j' = q_j + \eta \,(e_{ij} \,.\, p_i - \lambda q_j) \quad \text{..... (31f)}$$

$$b_i' = b_i + \eta \,(e_{ij} - \lambda b_i) \quad \text{..... (31g)}$$

$$c_j' = c_j + \eta \,(e_{ij} - \lambda c_j) \quad \text{..... (31h)}$$

The training algorithm is shown Algorithm 2.

Input: R: dataset,
η: learning rate,
λ: Regularization factor,
K: number of features.
Output: $\boldsymbol{U}^*$, $\boldsymbol{Q}^*$, $\boldsymbol{b}^*$, $\boldsymbol{c}^*$: the user and item feature matrices and biases,
n^*: the optimal number of epochs

1. Partition R into two sets: T(training set) 1 , V (validation set)
2. Initialize P, Q, b$*$ and c$*$ with small random numbers.
3. $n = 0$
4. **loop** until the terminal condition is met. One epoch:
5. $n = n + 1$
6. **for each** (i, j) element of T ordered by (u, d_{ij}):
7. $\hat{r}_{ij} = b_i + c_j + \sum_{k=1}^{K} u_{ik} q_{jk}$;
8. $e_{ij} = r_{\text{ij}} - \hat{r}_{ij}$;
9. /* update the variables according to eqs. $(\boldsymbol{viii}) - (\boldsymbol{xi})$ */
10. **for each** $k \in \{1, \ldots, K\}$
11. $u'_{ik} = u_{ik} + \eta \left(e_{ij} \,.\, q_{jk} - \lambda u_{ik}\right)$;
12. $q'_{jk} = q_{jk} + \eta \left(e_{ij} \,.\, p_{ik} - \lambda q_{jk}\right)$;
13. $u_{ik} = u'_{ik}$;
14. $q_{jk} = q'_{jk}$;
15. **end**
16. $b_i = b_i + \eta \left(e_{ij} - \lambda b_i\right)$
17. $c_j = c_j + \eta \left(e_{ij} - \lambda c_j\right)$
18. **end**
19. calculate the RMSE on V;
20. **if** the RMSE on V was better than in any previous epoch:
21. Let $\boldsymbol{P}^*$ = **P**, $\boldsymbol{Q}^*$ = **Q**, $\boldsymbol{b}^*$ = **b**, $\boldsymbol{c}^*$ = **c** and n^* = n;
22. **end**
23. terminal condition: RMSE on V does not decrease.
24. **end**

Algorithm 2: Training algorithm for BRISMF

2.5.1.1 Alternating least squares (ALS) for matrix factorization

Alternating least squares is a commonly used method in the field of Machine Learning to factorize matrices. It can be summarized as follows: first it initializes $\boldsymbol{Q}$ randomly, fixes that, and computes $\boldsymbol{U}$ using Ridge Regression (RR). Then it fixes $\boldsymbol{U}$, and computes $\boldsymbol{Q}$ using RR. These two steps are repeated until convergence, thus the algorithm alternates between recomputing $\boldsymbol{U}$ and $\boldsymbol{Q}$.

Because both $\boldsymbol{q_j}$ and $\boldsymbol{u_i}$ are unknowns, Equation 17 is not convex. However, if we fix one of the unknowns, the optimization problem becomes quadratic and can be solved optimally. Thus, ALS techniques rotate between fixing the $\boldsymbol{q_j'}$s and fixing the $\boldsymbol{u_i'}$s. When all $\boldsymbol{u_i'}$s are fixed, the system recomputes the $\boldsymbol{q_j'}$s by solving a least-squares problem, and vice versa. This ensures that each step decreases Equation 17 until convergence (Koren et al., 2010).

Description of the algorithm

The recomputation of $\boldsymbol{U}$ is performed by solving a separate least squares problem for each user. We can rewrite the following overall cost function (Istv´an Pil´aszy, 2009),

$$\sum_{(i,j)\in R} \left(\boldsymbol{r_{ij}} - \boldsymbol{u_i^T q_j}\right)^2 + \lambda \boldsymbol{u_i^T u_i} + \lambda \boldsymbol{q_j^T q_j} \quad \text{(32a)}$$

In the form:

$$\sum_{i=1}^{N} \left(\sum_{j:(i,j)\in R} \left(\boldsymbol{r_{ij}} - \boldsymbol{u_i^T q_j}\right)^2 + \lambda \boldsymbol{u_i^T u_i} + \lambda \boldsymbol{q_j^T q_j} \right) \quad \text{(32b)}$$

The outer sum is taken over each user. If $\boldsymbol{q_j'}$s are fixed, then only $\boldsymbol{u_i'}$s are to be optimized, which can be done separately for each user. For the $\boldsymbol{i}$ -th user, this leads to the following equivalent cost function:

$$\left(\sum_{j:(i,j)\in R} \left(\boldsymbol{r_{ij}} - \boldsymbol{u_i^T q_j}\right)^2 \right) + n_i \lambda \boldsymbol{u_i^T u_i} \quad \text{(32c)}$$

Where n_i is the number of ratings of user i, i.e. $n_i = \sum_{j:(i,j)\in R} 1$. Note that the constant term $\lambda q_j^T q_j$ is removed. This is a least squares problem without any linear inequality constraints, which can be solved by ridge regression as follows: let the matrix $\boldsymbol{Q}[u] \in \mathbb{R}^{n_i \times K}$ denote the restriction of **Q** to the items rated by user $\boldsymbol{i}$, the vector $\boldsymbol{r_i} \in \mathbb{R}^{n_i \times 1}$ denote the ratings given by the $\boldsymbol{i}-$th user to the corresponding items, and let :

$$\boldsymbol{A_i} = \boldsymbol{Q}[u]^T \boldsymbol{Q}[u] = \sum_{j:(i,j)\in R} \boldsymbol{q_j} \boldsymbol{q_j^T} \quad \text{..... (33a)}$$

$$\boldsymbol{d_i} = \boldsymbol{Q}[u]^T \boldsymbol{r_i} = \sum_{j:(i,j)\in R} \boldsymbol{r_{ij}} \cdot \boldsymbol{q_j} \quad \text{..... (33b)}$$

Then ridge regression recomputes $\boldsymbol{u_i}$ as:

$$\boldsymbol{u_i} = (\boldsymbol{\lambda} n_i \mathbf{I} + \boldsymbol{A_i})^{-1} \boldsymbol{d_i} \quad \text{..... (33c)}$$

After recomputing all $\boldsymbol{u_i}$s, the algorithm recomputes all $\boldsymbol{q_j}$s, using equations 33a and 33b, but interchanging the role of users and items.

While in general stochastic gradient descent is easier and faster than ALS, ALS is favorable in at least two cases. The first is when the system can use parallelization. In ALS, the system computes each $\boldsymbol{q_j}$ independently of the other item factors and computes each $\boldsymbol{u_i}$ independently of the other user factors. This gives rise to potentially massive parallelization of the algorithm. The second case is for systems centered on implicit data. Because the training set cannot be considered sparse, looping over each single training case, as gradient descent does, would not be practical. ALS can efficiently handle such cases (Koren et al., 2010).

2.5 The SVD++ Algorithm

In accounting for the user and item effects that are typical of CF data – i.e., systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others, (Koren Y., 2008) encapsulates these effects within the *baseline estimates* models. Denoting the overall average rating by $\boldsymbol{\mu}$, a baseline estimate for an unknown rating $\boldsymbol{r_{ij}}$ is denoted by $\boldsymbol{b_{ij}}$

$$\boldsymbol{b_{ij}} = \boldsymbol{\mu} + \boldsymbol{b_i} + \boldsymbol{b_j} \quad \text{..... (34)}$$

Where $\boldsymbol{b_i}$ and $\boldsymbol{b_j}$ represent the observed deviations user $\boldsymbol{i}$'s and item $\boldsymbol{j}$'s ratings from the average.

In order to estimate $\boldsymbol{b_i}$ and $\boldsymbol{b_j}$ one can solve the least squares problem:

$$\min_{b_*} \sum_{(i,j)\in K} \left(\boldsymbol{r_{ij}} - \boldsymbol{\mu} - \boldsymbol{b_i} - \boldsymbol{b_j}\right)^2 + \boldsymbol{\lambda_1}\left(\sum_i \boldsymbol{b_i^2} + \sum_j \boldsymbol{b_j^2}\right) \quad \text{(35)}$$

The first term $\sum_{(i,j)\in K} \left(\boldsymbol{r_{ij}} - \boldsymbol{\mu} - \boldsymbol{b_i} - \boldsymbol{b_j}\right)^2$ strives to find $\boldsymbol{b_i}$'s and $\boldsymbol{b_j}$'s that fit the given ratings. The regularizing term $\boldsymbol{\lambda_1}\left(\sum_i \boldsymbol{b_i^2} + \sum_j \boldsymbol{b_j^2}\right)$ avoids overfitting by penalizing the magnitudes of the parameters.

Applying SVD in the CF domain raises difficulties due to the high portion of missing ratings. Conventional SVD is undefined when knowledge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known entries is highly prone to overfitting. Earlier works relied on imputation to fill in missing ratings and make the rating matrix dense. However, imputation can be very expensive as it significantly increases the amount of data. In addition, the data may be considerably distorted due to inaccurate imputation. Hence, more recent works suggested modeling directly only the observed ratings, while avoiding overfitting through an adequate regularized model, such as:

$$u_*, \min_{q_*, b_*} \sum_{(i,j)\in K} \left(r_{ij} - \mu - b_i - b_j - u_i^T q_j\right)^2 + \lambda_2\left(\|u_i\|^2 + \|q_i\|^2 + b_i^2 + b_j^2\right) \quad \text{(36)}$$

A simple gradient descent technique as in Algorithm 2; can successfully be applied to solve equation (36).

In integrating implicit feedback, (Koren Y., 2008) suggests a direct modification of a generalized predictive form of (27) expressed as:

$$\hat{r}_{ij} = b_{ij} + u_i^T q_j \quad \text{(37)}$$

The improved prediction would therefore be thus written:

$$\hat{r}_{ij} = b_{ij} + q_j^T \left(u_i + |\mathrm{N}(i)|^{-\frac{1}{2}} \sum_{g \in \mathrm{N}(i)} y_g \right) \quad \text{(38)}$$

Now a user $\boldsymbol{u}$ **is** modeled as $\boldsymbol{u_i + |\mathbf{N}(i)|^{-\frac{1}{2}} \sum_{g \in \mathbf{N}(i)} y_g}$**.** This applies a free user-factors vector, $\boldsymbol{u_i}$**,** which is learnt from the given explicit ratings. This vector is complemented by the sum $\boldsymbol{|\mathbf{N}(i)|^{-\frac{1}{2}} \sum_{g \in \mathbf{N}(i)} y_g}$, which represents the perspective of implicit feedback. This model is hence referred to as the SVD++. Model parameters are learnt by minimizing the associated squared error function through gradient descent as applied in Algorithm 2.

2.6 Conclusion

From this discussion, we noticed that Recommendation Systems (RS) are inherently based on machine learning techniques of linear regression, neural networks and other linear algebraic models. The key point of observation is that these models require training; usually done using training sets from the data. During training, a learning rate is determined and set, and then an algorithm such as the Least Squares or Gradient Descent are applied to minimise a cost function. Common measures used for the cost function are the MSE or the RMSE. At a point of convergence, the algorithm learns the weights of the hypothesis used and applies these in subsequent future predictions.

This thesis employs the gradient descent algorithm on a Linear Regression model that learns the weights of each tweet factors to be able to predict future user absence durations. The temporal absence thus predicted from this model can be recommended to peers in a Social Networking systems or otherwise determine what items to recommend in an e-commerce application.

CHAPTER THREE

3.0 RESEARCH METHODOLOGY

This chapter elaborates the research design and the specific approach that was adopted in this study to investigating the appropriate temporal absence modeling techniques. The chapter describes the methodologies employed and their application within the study. We begin by discussing the research design used in the study and the basis for its adoption; then we discuss the data collection process, the sources of the data, sampling, and the data itself; after which we describe the analysis process and the results.

3.1 Research Design

This research took two different approaches: initially, there was a case based study of existing recommendation data that helped determine the existence of Temporal Absence. The chosen case study was the Netflix prize data that had been used in the KDD cup competition, a further elaboration of this is made in the next section. An empirical research design approach was then adopted, in which we sought to investigate the algorithms and models used in CF – based recommendation systems and the techniques used to model time dynamics; this helped capture the methods that can be used to model temporal absence patterns. With the techniques so found, an inherently correlation study approach is adopted where variables in Temporal Absence patterns was studied in a bid to discover their effect on the recommendation results of the algorithms, this was then later modeled for use in recommendation algorithms.

3.2 Data collection

Due to the research approach, two sets of data were employed to be able to accomplish the aim of this study. First, Secondary data obtained from recommender systems that have been collected

over time. This data is vital since it consists of tried and tested data that has been used in successful research in this field and consisted of publicly available datasets. Such datasets have been extensively used for example (Rashid et al., 2005) employs data from www.grouplens.org.

For this research, we employ the Netflix prize data, majorly because it is the most widely studied dataset in recommendation systems and perhaps even the data that defined this new area of research.

Secondly, we collected data from social media users through a live questionnaire administered online through Google docs. This set of data is used to determine the factors and aspects of social media temporal dynamics and hence quantify the same. The results of the analysis of this data are used to develop the prediction model described in chapter 5.

3.2.1 The Netflix dataset

In 2006, the online DVD rental company Netflix announced a contest to improve the state of its recommender system. To enable this, the company released a training set of more than 100 million ratings spanning about 500,000 anonymous customers and their ratings on more than 17,000 movies, each movie being rated on a scale of 1 to 5 stars. Participating teams submitted predicted ratings for a test set of approximately 3 million ratings, and Netflix calculates a root-mean- square error (RMSE) based on the held-out truth. The first team that would improve on the Netflix algorithm's RMSE performance by 10 percent or more would win a $1 million prize. Had no team reached the 10 percent goal, Netflix would give a $50,000 progress prize to the team in first place after each year of the competition (Koren et al., 2009).

This dataset has been employed by nearly all research since that period to date since it is the largest existing dataset known since the history of recommender systems. In addition to its size

and availability, the dataset is 4 tuple with each entry consisting of {user id, item id, rating, and time}, the time attribute is normally absent in other datasets like the movielens dataset. This follows that it is the most credible, proven dataset in researches involving recommendation.

3.2.2 Social Media Data.

As stated earlier, a questionnaire, included in Appendix III, was administered online through Google docs targeting users of social media sites. The choice of social media for this study was due to the fact that social media recommendations are becoming increasingly popular majorly since social media itself has become a part of day to day life activity. It is something close to a phenomenon and all users of online systems find themselves in one way or another involved in social media.

Social Media has therefore transformed from sites of sharing social attributes of life into a market palace containing a plethora of items ranging from information, education materials, news feeds and social chats to ecommerce activities like marketing, advertising and many more. It is no wonder therefore that recommendation systems have crept into Social Media and as a matter of fact these recommendations are assisting the users thereof to filter out the necessary from unnecessary activities in social media. A notable social media recommendation systems in the “people you may know” and “pages you may like” recommendations that seek to suggest to a user, some friends or people they may be interested in based on their profiles and activities.

We took a random sample of social media users since this would give a good picture of overall use patterns from the very frequent users to less enthusiastic ones, (Lindel et al, 2003) has employed this sampling technique to solve scaling issues of the recommendation algorithms. For this research, the sampling required to random so as to provide a wholesome and indiscriminate

composition of the data. From such a sample, we were sure to study trend of active users together more irregular ones.

CHAPTER 4

ANALYSIS AND RESULTS

4.1 Temporal Absence Connotations in Netflix Data

To study the Temporal Absence from the Netflix data; given that this is a very large dataset of over three (3) million ratings spanning a period of over six (6) years, we divide this time into time bins, a concept discussed by (Koren Y., 2009) that creates smaller time windows from the larger period. The unique customer ids within each time bin were then obtained and subsequently mapped into a graph of the same. This provides a view of presence patterns within the system. The presence graph is shown in figure 5.

4.1.1 Binning

We use a bin size of 10 days resulting into two hundred and twenty five (225) time bins for this study but the 225th bin has only two important days so it was discarded to leave 224 effective time bins; table1 below indicates the binning periods sample for the first twenty (20) time bins. It should be noted that the last time bin period is 2005-12-19 – 2005-12-28.

Table 1: First 20 10-day bin periods

#BIN	PERIOD	#BIN	PERIOD
bin1	1999-11-11 - 1999-11-20	bin11	2000-02-19 - 2000-02-28
bin2	1999-11-21 - 1999-11-30	bin12	2000-02-29 - 2000-03-09
bin3	1999-12-01 - 1999-12-10	bin13	2000-03-10 - 2000-03-19
bin4	1999-12-11 - 1999-12-20	bin14	2000-03-20 - 2000-03-29
bin5	1999-12-21 - 1999-12-30	bin15	2000-03-30 - 2000-04-08
bin6	1999-12-31 - 2000-01-09	bin16	2000-04-09 - 2000-04-18
bin7	2000-01-10 - 2000-01-19	bin17	2000-04-19 - 2000-04-28
bin8	2000-01-20 - 2000-01-29	bin18	2000-04-29 - 2000-05-08
bin9	2000-01-30 - 2000-02-08	bin19	2000-05-09 - 2000-05-18
bin10	2000-02-09 - 2000-02-18	bin20	2000-05-19 - 2000-05-28

4.1.2 Absence Connotations

Since the Netflix data does not come with the customers' registration information; we are only provided with the movies and rating information, yet our interest for this study is more inclined to the customers/users of the system, it becomes a challenge to know exactly how many customers were not present from the system. To circumvent this problem we use the customer id fields to get the low bound number of customers present in the system i.e. since the customer id is auto incremented from 1, we can confidently tell that if for bin X, the highest customer id is a given figure Y, and then the least number of customers present within the system at this bin time is Y. Subsequently, for any successive bin, if there is no higher customer id than Y, then Y remains the least number of customers within the system for that bin, otherwise we update Y with the new highest customer id from the bin. $P_{LB,Y} = \{\max(\text{customer}_{id}, \text{Bin}_{1\ldots Y})\}$

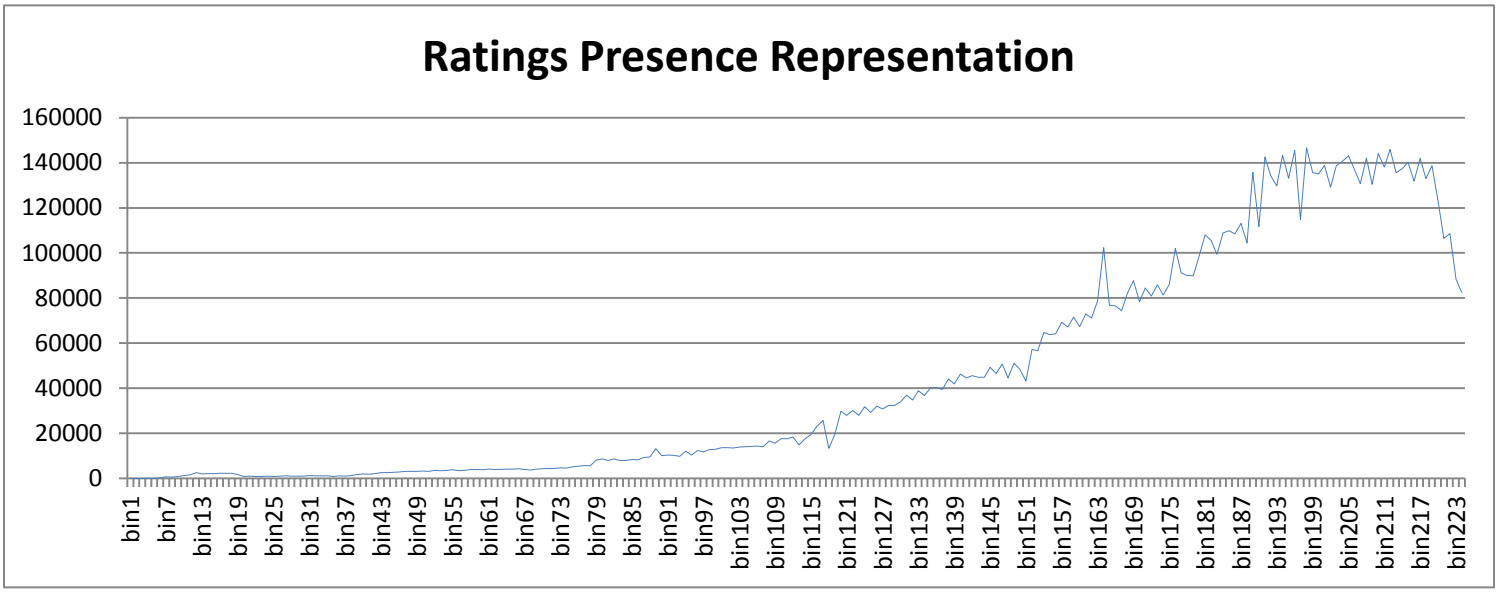


Figure 4: Bin ratings presence graph

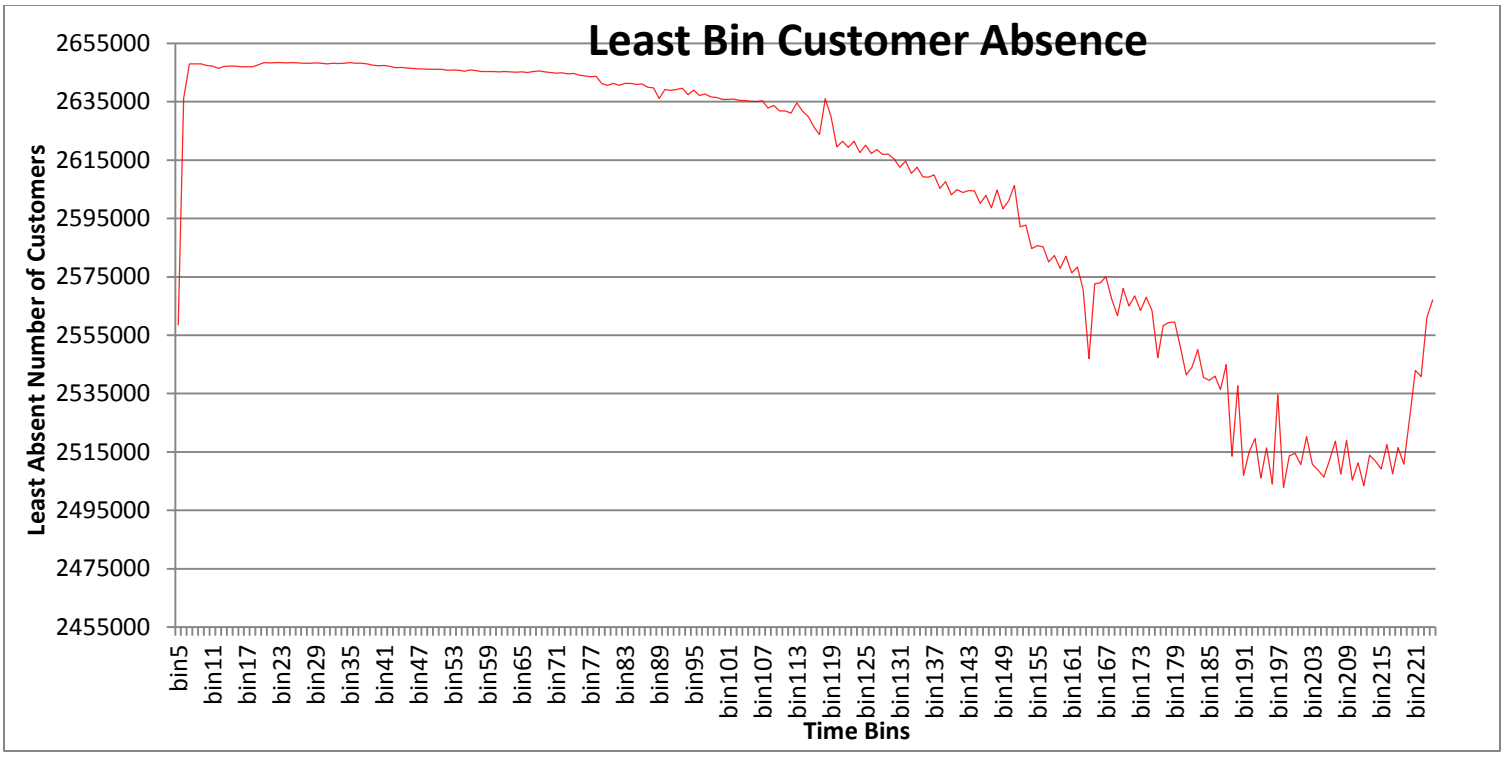


Figure 5: Least Bin Customer Absence graph

The graph in figure 4 shows the variations in the actual numbers of customers present in the Netflix systems within each 10-day bin period, figure 5 on the other hand shows the minimum absentees from the systems in every time bin. A comparison of these graphs indicates the disparity of numbers between temporally absent users and present; where it is visible that the actual users within the systems at different time bins are way less than those that are absent. Given that the numbers graphed in figure 5 are not the exact absentees due to the nature of the data and that the actual figures would be even larger, it is clear that giving weight to temporal absence is key in recommendations. At this point, the key questions that ring in mind would include: what caused this absence, what could be changed in the systems to reduce these absence numbers, and whether or not an algorithm should exclude these figures from calculations. From the two graphs, the first ten bins do not directly correlate and the graphs in figure 6 explain this.

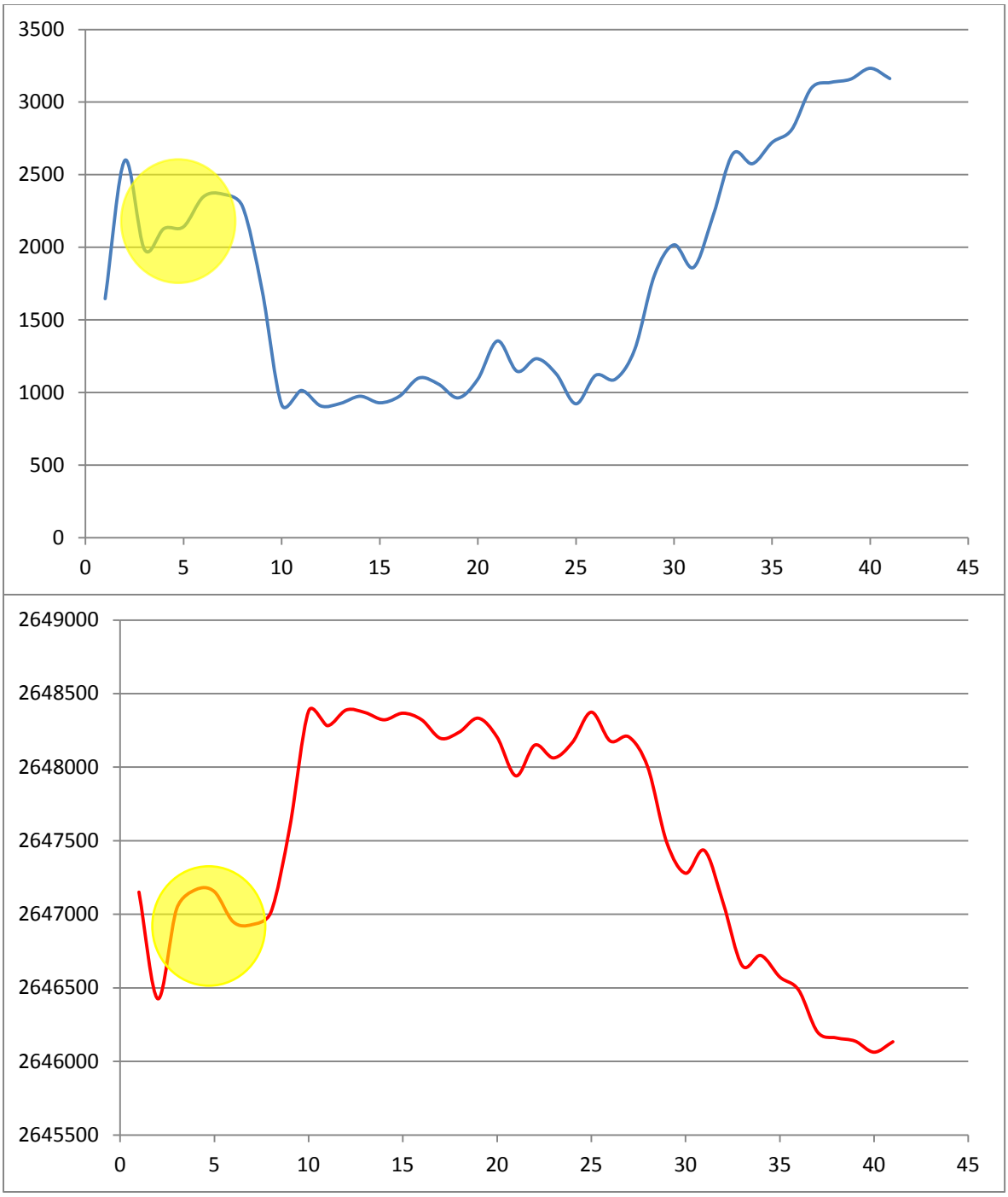


Figure 6: Windowed comparison of absence/present to reveal finer shifts in customer numbers

From the yellow circle points, we note that the number of customers in the system change with time; in most cases within this data, the highest number of rating users remains constant but at some points it shifts as indicated by the bins between bin3 to bin10 as opposed to the previous

overall graphs that hid such fine details. It is therefore prudent to assume that the overall numbers in the system will constantly be changing hence the number of absent uses of the ratings system is not a mere reflection of the absence based on some fixed total number of users.

4.1.3 Absence Modeling for E-Commerce

From the above representations, it clear that the numbers that are absent from the system at any given bin period way exceeds the numbers present. Given that the actually present individuals in the system or those that actually do the ratings at any given instance of time change from bin to bin, it will be prudent to determine which particular users would be present in a bin so that during the algorithmic run, we have a highly reduced number of users to work with. Such is the modeling that is required to be achieved to attain efficiency and accuracy from temporal absence connotations in the data.

To achieve this, the following steps should be taken:

1. Determination of each user's absence pattern from the rating system; here it's important to note that the given user might actually be present in the system or otherwise be signed in to the system but not actively involved in system ratings or for the Netflix case, purchase of movies. To achieve this, a perceptron or a regression analysis algorithm could be used; the end result of which is to have a separation of which users are temporally absent from the ratings system and during which time bins.
2. The results of step one would then be fed into the SVD++ algorithm in a bid to predict future absence patterns. This would mean that the information fed to the algorithms would have to be reconstructed into a suitable low rank matrix.

3. The result of the above step gives us a discrimination criterion for selecting which particular user will be required for a given instance run of the algorithm. Since the number of users actually present in the system within any bin period is actually way less than the actual population of the user – item matrix, this will be very cost effective in terms of execution time.
4. The last step of the improved algorithm would be to determine whether a user who has been absent from the system should be considered temporarily absent or is s/he within their absence patterns, is considered temporally absent then the mode of operation of the algorithm is altered to encompass the absence factor but if otherwise found to be within their absence patterns, their ratings predictions would be computed as usual.

These steps form the four major steps of a temporal absence model of an e-commerce recommendation algorithms, any developer of such algorithm should seek to address the given steps during design.

4.2 Social Media Data Analysis

The analysis of the data obtained from the social media questionnaire aims to identify two aspects of absence:

i. Absence latent factors: Weight for predicting absence.

ii. Absence indicators: Covariates for absence.

The Absence Latent Factors are events or occurrences within the system or without that affect the user's presence in the system, they may not be termed directly as causes of absence since most of them just partially contribute to the absence and may not be directly quantifiable in terms of their contribution. Some of these events actually affect other factors which in turn then lead to

the absence; a good example being political affiliations. Take for instance a user who simply dislikes politics; it will follow that during electioneering periods, this user would be expected to be at least passive if not absence in totality from the system. And so in general the algorithm can be molded to give weights to political periods.

On the other hand, this factor can be viewed as only affecting a user's interest in the system, a major factor that will become later known as diminishing interest, which in turn leads to temporal absence. If taken in this sense, political affiliation becomes a minor sub factor that simply contributes in a small extent to the overall absence, in this perspective then we see that other factors must come into play for us to even give good weight to politics, these factors may include current topic of discussion or even discrediting friends.

The question of how much weight we give to politics in this second perspective is dependent on whether the political environment in the real world or in a country has translated into a political social media space; if this maps directly and politics becomes a "hot" topic of discussion in social media then the user is likely to alter their behavior either positively or negatively. The debate is therefore down to whether the user's environment and profile is translated into social media space. It is also important to note that a factor like politics may not really be the major topic of current discussion but can easily form a source of social media bullying. For most users, social media bullying, intimidations from friends and discrediting friends are a common factor for withdrawal from social media. Discrediting friends can easily use political affiliations to reduce a user's motivation. Political factors therefore can be viewed from two directions one as a factor that can cause absence on its own or otherwise as a factor that relies on other factors or contributes to another major factor.

Absence indicators are visible results of latent factors. They are behavioral changes in social media that can be quantified to indicate a user was absent or will be absent. At this point we make a note that absence indicators can either indicate when a user is nearing an absence period or that a user has just emerged from an absence period both of which are vital to a recommendation system. In the former sense, absence indicators are weighted to be able to predict when and for how long the user will be out from the system but the later forms what becomes a user's behavior after being out of the system, for e-commerce systems, this is very vital since then we can know whether there exists a concept drift since the last period of activity.

Before we get to discuss these two aspects of recommendations, we draw attention to the existing forms of recommendations and what the user thinks about the same. We also make a note on how evaluation of recommendations would depend on these factors.

4.2.1 Existing Recommendations

One major social media recommendation is the people you may know recommendation presented in virtually all social media sites like Facebook, Twitter and G+. This is the form of discovery of new friends for social media users. The graph in figure 7 shows the perception of social media users on the effectiveness of these recommendations.

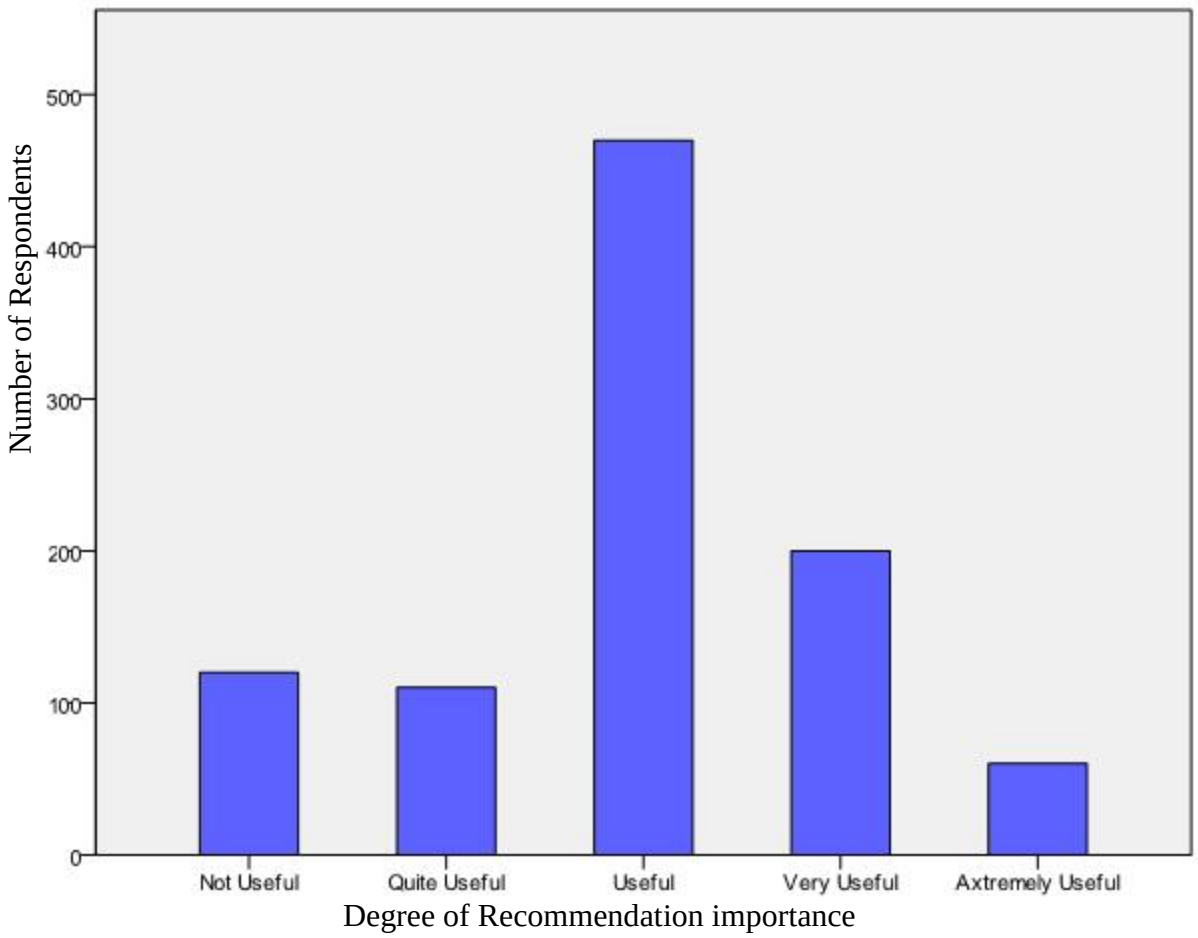


Figure 7: Graph of usefulness of social media friends' suggestions

Useful Vs. Relevant Recommendations

Most respondents found the recommendations useful translating to the fact that they assist in discovery of new friends. We then seek to know how relevant the recommendations are i.e. in a list of ten friends suggestions, how many are actually important and how many would be irrelevant. The general analysis brings out that idea that most of these friend suggestions are irrelevant recommendations meaning that only about 20% of all friend recommendations are exactly what the user would require. The rest would probably be people they don't know or people they don't want

to befriend; the key question here is why you would not want to befriend someone if you knew them and some of the factors under consideration in answering this question include:

- ***Too old friends*:** Some users think that some of the friends suggested are people they met too long a time ago that there is probably ideological or status gap that they probably might not need to be friends anymore. Take for example the world's great footballer Christiano Ronaldo born in Medeira (a small island in Portugal) where he grew up playing football with his local friends; at the age of 12, he leaves his birth place for better opportunities in England and later Spain from where he wins his second Ballon d'or at the age of 28. When commenting about this achievement, his childhood friend Rui says:

  "People don't know him. They think they do but when you are with him, he is such a kind and generous person. ***He doesn't come back here [to Santo Antonio] because many of these guys don't work and ask him for money, so he stays away****, but he is still the same person" (Goal.com, URL-1)*

  This behavior of staying away from people who were once close friends in life is a common social behavior of human beings and it reflects in social media too. A significant number of respondents feel that this is a factor that makes some friends suggestions irrelevant.

- ***Difference in Social Life*:** Many a user of the social media sites would first seek to view the profile of a recommended individual and comprehend how they behave in social media space. Some users might hold a strong religious stand on issues yet one of their friends is a friend to another who practices religiously controversial practices albeit accepted in the society like use of drugs, pornography, public adultery, homosexuality etc. It is more than

likely that such a friend would be suggested and that the user in question would find it irrelevant.

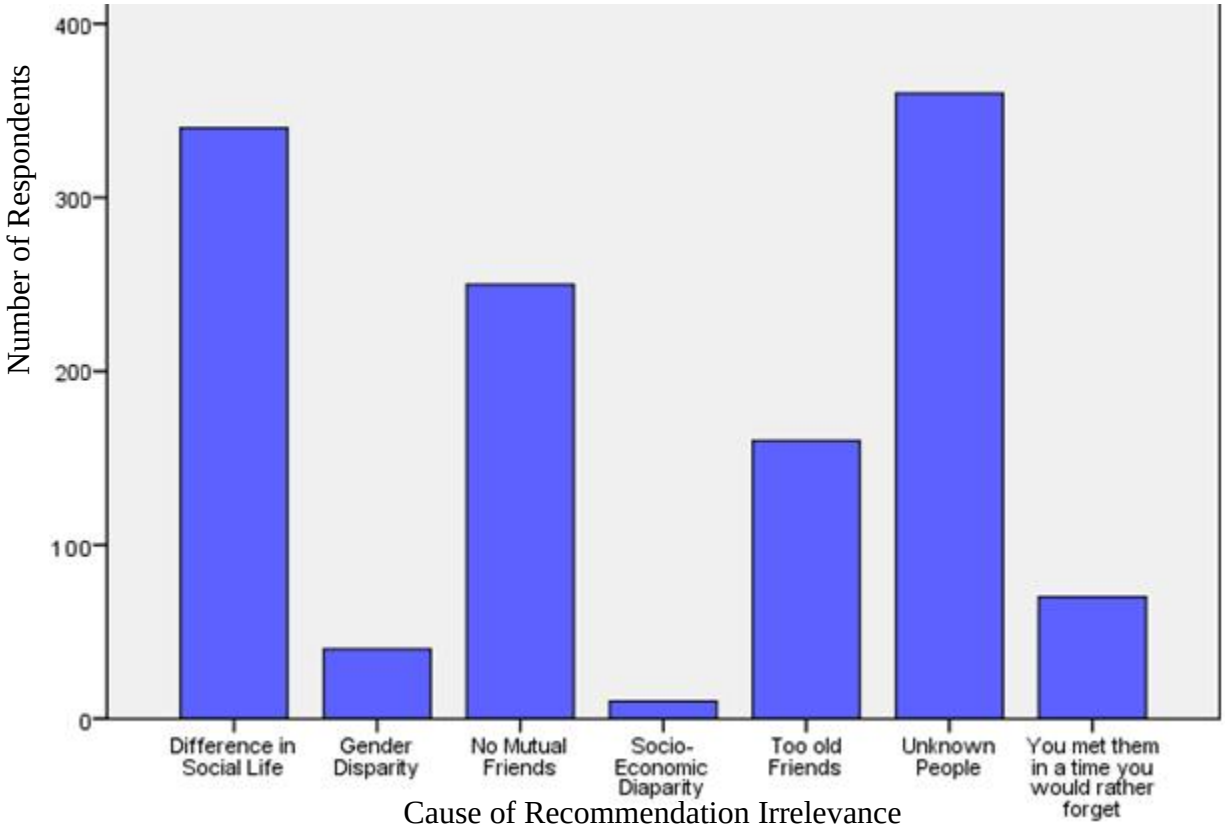


Figure 8: Causes of Recommendation Irrelevance.

4.2.2 Recommendation Quality for User Centric evaluation

From the foregoing discussion, it's important that we can look at how far research on factors affecting recommendation acceptance by users has gone. Researchers currently consider a given array of factors for user – centric evaluation. In social media recommendations, these factors are affected by the temporal aspects discussed before under section 4.2.3.

a) Quality of Recommended Items

i) Diversity

As discussed by (Lathia N. et al., 2010), diversity is the ratio between the new items in the current list to the total number of items in the list. As the recommendation list is the first piece of information users will encounter before they examine the details of an individual recommendation, users' impression of this list is important for their perception of the whole system. (Peral Pu & Li Chen, 2010).

ii) Novelty

A list's novelty will be high if all of the items have never been recommended before, and low if all of the items have been recommended at some point in the past (Lathia N. et al., 2010). Novelty can also be viewed as discovery (Pearl Pu and Li Chen, 2010) and defines the recommender's ability to educate the users and help them discover new items. (Herlocker et al., 2004), argued that novelty is different from a new concept: "serendipity", because novelty only covers the concept of "new" while serendipity means not only "new" but also "surprising" but this distinction might cause confusion in actual evaluation procedures.

iii) Perceived Accuracy

This refers to the degree to which users feel the recommendations match their interests and preferences. It is an overall assessment of how well the recommender has understood the users' preferences and tastes (Chen L. and Pu P., Sep, 2010) which is significantly easier to obtain than the measure of objective accuracy (Chen L. and Pu P., Feb, 2010).

iv) Familiarity

A description of whether or not users have previous knowledge of, or experience with, the items recommended to them. Although (Swearingen and shinha, 2002) note that recommendation of previously experienced items would reinforce users' trust in the system, there needs to exist a balance between familiarity and novelty.

v) Attractiveness

Refers to whether or not recommended items are capable of stimulating users' imagination and evoking a positive emotion of interest or desire. While judging novelty requires a user to think more about the distinguishing factors of an item, the aspect of attractiveness brings to mind the outstanding quality of an item and has a more emotional tone to it (Chen L. and Pu P., Sep, 2010).

vi) Enjoyability

Found to have a significant correlation to users' intention to use and return to the system, it refers to whether users have enjoyed experiencing the items suggested to them. In many online study scenarios, it is not possible to immediately measure enjoyability unless users are told to answer a questionnaire after a few weeks when they have actually received and experienced the item.

vii) Context Compatibility

This is an evaluation of whether or not the recommendations consider general or personal context requirements. A good recommender system should be able to formulate recommendations considering different kinds of contextual factors that will likely take effect.

b) Interaction Adequacy

Besides issues related to the quality of recommended items, the system's ability to present recommendations, to allow for user feedback and to explain the reasons why recommendations facilitate purchasing decisions also weighs highly on users' overall perception of a recommender (Pearl Pu and Li Chen, 2010). Three main interaction mechanisms are usually suggested in various recommenders: initial preference elicitation, preference revision, and the system's ability to explain its results.

Behavioral based recommenders do not require users to explicitly indicate their preferences, but collect such information via users' browsing and purchasing history. For rating and preference based recommenders, this process requires a user to rate a set of items or state their preferences on desired items in a graphical user interface. Some conversational recommenders provide explicit mechanisms for users to provide feedback in the form of critiques. The simplest critiques indicate whether the recommended item is good or bad, while the more sophisticated ones show users a set of alternative items that take into account users' desire for these items and the potential superior values they offer, such as better price, more popularity, etc.

The final interaction quality being measured is the system's ability to explain the recommended results. A good explanation interface could help inspire users' trust and satisfaction by giving them information to personally justify recommendations, increasing user involvement and educating users on the internal logic of the system. In addition, (Tintarev and Masthoff, 2007), defined in detail possible aims of explanation facilities: transparency, scrutability, trust, effectiveness, persuasiveness, efficiency, and satisfaction.

c) Interface Adequacy

Work on the user interface and layout for presenting the Recommendations has concentrated on how to optimize the recommender page layout to achieve maximum visibility of the recommendation. User interface factors such as information sufficiency, the interface label and layout adequacy and clarity should be considered (Pearl Pu and Li Chen, 2010).

In conclusion, the evaluation of current "*people you may know*" or "*pages you may like*" recommendations on social media which are inherently based on Neighborhood alone, would determine these recommendations as poor given the results shown in the graphs in figures 7 and 8.

4.2.3 Absence Latent Factors

Section 4.2.2 shows ineffectiveness of current recommendations as being caused by factors that are essentially related to time and user aspects; temporal modeling can show changes in a user's preferences which is commonly known as concept drift. The latent factors are the underlying courses of the absence and we foster here that poor recommendations or simply changes in a user's interest are major causes of absence of users from the system.

In e-commerce recommendations the MF algorithms consider a given range of latent factors or features. The K features in the algorithm are directly dependent on the latent factors that are observed from the data (Istv´an Pil´aszy, 2009). These features are not observed directly but are some underlying factors that would affect a user's rating in the system.

From the Administered Questionnaire; I determine these latent factors that could lead to absence from social media as follows:

i) Ill health

This is a natural occurrence factor that cannot be measured within the system and hence cannot be quantified but is a latent factor and a cause of absence. The graph below shows how ill health affects presence of users within the system.

ii) Job schedules

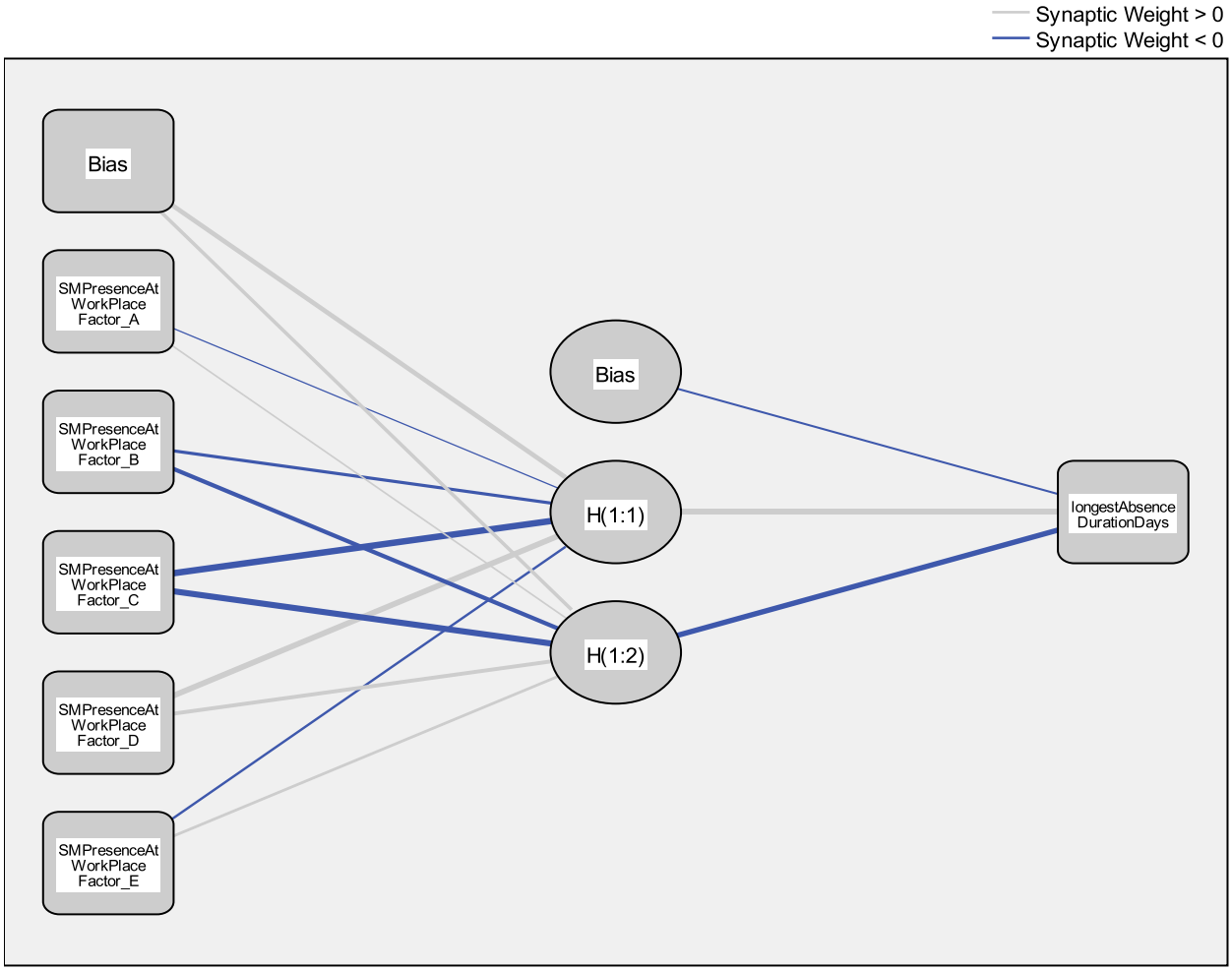
Job schedules is a multifaceted factor that can influence a user's presence in social media, fixed working hours might not directly translate to busy schedules and in any case these are immeasurable quantities on social media unless there is *continuous user profiling* which then leads us to a tradeoff between the length and frequency of user profiling as discussed by (Chen Li and Pu Pearl, Fe, 2010 & Lathia N. et al., 2010) We identify five factors in the work place that affects job schedules and ultimately affect social media presence or absence as listed below:

- Busy work schedules: tokenized work where there is a measured work quantity for the user during official working hours.
- Disallowed Social Media at work places: In some instances, the organization to which the user belongs blocks social media sites during office hours, this would mean that the user is expected to be absent from social media even if they were idle in the work place. For users that prefer using computers more than hand held devices for access, this would be an absolute hindrance.
- Travelling and movement to and at the work place: This would have the same effect to travelling as discussed part of section 4.2.3.
- Task deadlines: In a project based work environment in which each project/task has a measurable time schedule and milestones to be reached, the proximity of these target points would affect a user's presence on social media. A frequent user might take

near three weeks absence to be able to meet their deadlines. This factor might be very difficult to monitor and may only be achieved through *continuous profiling* or the user's social media content and text analysis.

- Lack of internet: Remote areas with little or no internet connection or an organizational policy to deny user's internet connection may result in absence of a user. These may happen in high integrity offices with highly sensitive information.

Figures 9, and 10, show how these factors fit into a multilayer perceptron neural network to reveal their effects on the user's absence. We observed the users' longest duration of absence from social media and this is our targeted value for prediction: The network is trained by the independent variables from the above discussion and known absence durations. Finally a probe set is used to test the model with a sum of square error of 24.56.



Hidden layer activation function: Hyperbolic tangent

Output layer activation function: Identity

Model Summary

Training	Sum of Squares Error	15.261
	Relative Error	.610
	Stopping Rule Used	1 consecutive step(s) with no decrease in error[a]
	Training Time	0:00:00.031
Testing	Sum of Squares Error	24.561
	Relative Error	.909

Dependent Variable: Longest Duration of Absence from Social Media in Days

a. Error computations are based on the testing sample.

Figure 9: Multilayer Perceptron for Effect of Workplace Factors on Absence

Table 2: Work Place Multilayer Perceptron Network Information

Network Information

Input Layer	Covariates	1	Degree of Effect of Busy Work Schedule on Respondent's Absence
		2	Degree of Effect of disallowed Social media at work place on Respondent's Absence
		3	Degree of Effect of Travelling to work stations on Respondent's Absence
		4	Degree of Effect of Deadlines at work on Respondent's Absence
		5	Degree of Effect of Lack of internet at work on Respondent's Absence
		Number of Units[a]	5
		Rescaling Method for Covariates	Standardized
Hidden Layer(s)		Number of Hidden Layers	1
		Number of Units in Hidden Layer 1[a]	2
		Activation Function	Hyperbolic tangent
Output Layer	Dependent Variables	1	Longest Duration of Absence from Social Media in Days
	Number of Units		1
	Rescaling Method for Scale Dependents		Standardized
	Activation Function		Identity
	Error Function		Sum of Squares

a. Excluding the bias unit

iii) Current topic of discussion

Topics of discussions change from time to time in social media and this affects users' engagement, A user who for example dislike political issues, will be less available during electioneering periods. From the data, the major topics that affect user's absence include:

- Politics : Political figures, parties, governments
- Personality discussions
- Ethnicity
- Religion and Spiritualism: Denominations, Scripture, Illuminati e.t.c
- Sports: Football, sports personalities, Teams

iv) Travelling

Travelling affects a very small number of social media users but in this rare case, these users are adversely affected and might be absent from social media. When travelling either the user will change their mode of access to mobile or simple not be present in social media at all. From our data only 9% of users acknowledge that they are affected by travelling.

v) Change of access device (e.g. from mobile to desktop or vise vasa).

Change of devices of access can easily lead to diminishing interest as discussed in part viii of this section; Users who are compelled to alter their access could lose the interest all together.

vi) Discrediting friends

Social media bullying is a well-documented problem on social networks with some states even enacting laws to govern and curb it. Discrediting friends in social media could easily lead to cyber bullying. Emotionally weak users will be adversely affected by discrediting

friends and might lead to total absence from the user. There is though, a level of discrediting that is not really bullying but is strong enough to trigger a withdrawal from users.

vii) None Responsive friends

These exist mainly in chat engine based social media Networks such as Facebook. From the response obtained via our survey; non responsive friends cause negative emotions on a user which may in turn lead to absence from the system.

viii) Diminishing Interest

Diminishing interest is an interesting factor since it depends on other factors including the ones already discussed in parts i to vii of this section but above that, it directly leads to temporal absence; to observe the relationship between these contributing factors and their contribution to diminishing interest of a user, we fit these factors and predict a user's diminishing interest through a Multilayer Perceptron Neural Network, figure 10, that determines whether a user is experiencing diminishing interest or not. This is therefore a classification problem that trains the network based on the values for the factors above and known classifications. Table 3 shows the factors that affect a user's diminishing interest while table 4 shows a summary of the network performance on the training and test data; and the error rates.

Table 3: Factors affecting diminishing interest

Multiple Response Sets

Name	Label	Coded As	Counted Value	Data Type	Elementary Variables
Causes of Diminishing Interest	Diminishing interest Aspects	Categories	N/A	Numeric	Busy Schedule Travelling Nagging Social Media Friends Lack of Response from Friends Nothing to do online Lack of Internet Connection Poor Health Not liking existing conversations

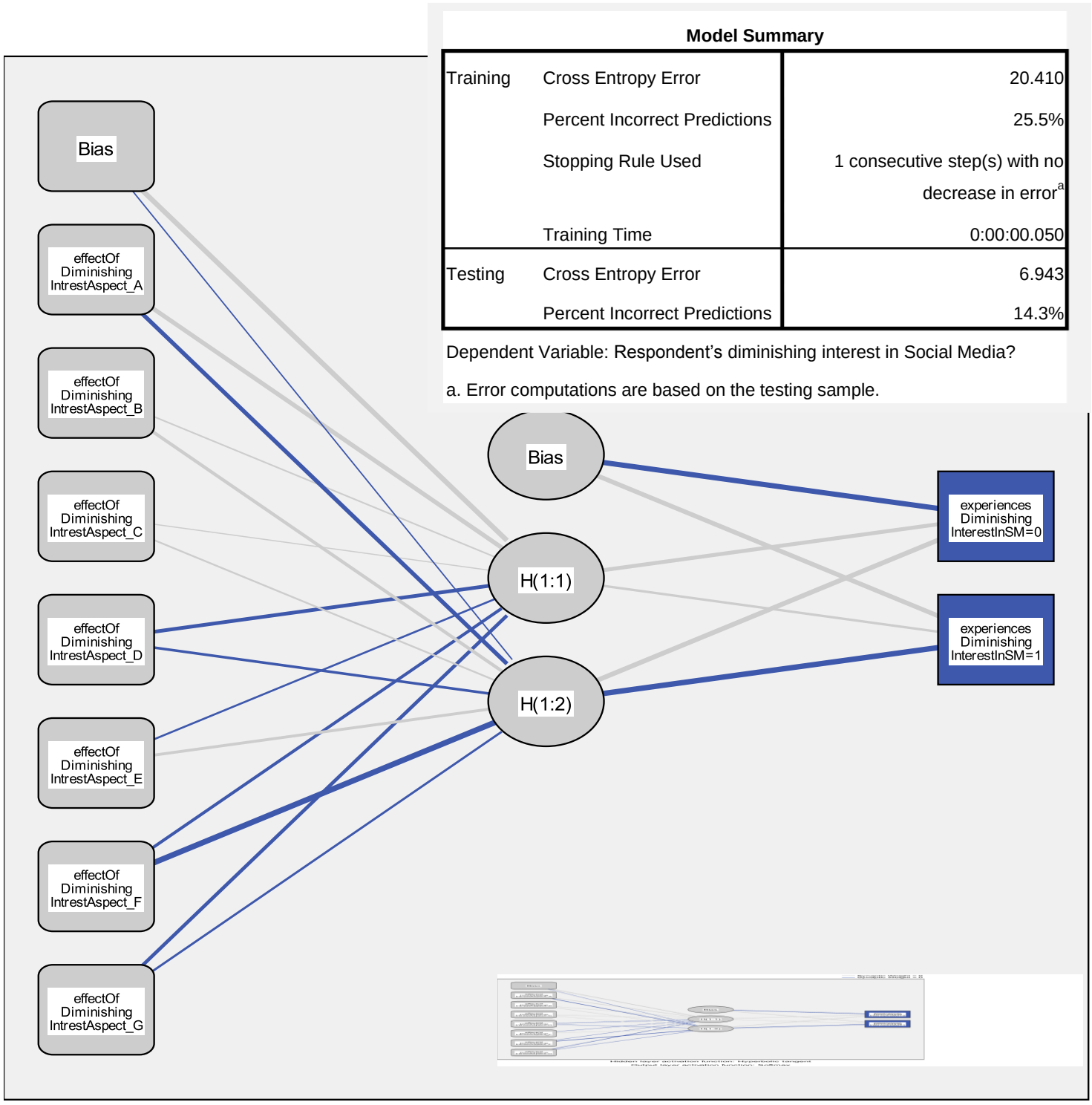


Figure 10: Multilayer Perceptron Classifier for Diminishing Interest

Table 4: Diminishing Interest Classification Accuracy

Classification

Sample	Observed	Predicted		
		No	Yes	Percent Correct
Training	No	541	449	54.5%
	Yes	198	792	80.0%
	Overall Percent	27.5%	72.5%	74.5%
Testing	No	149	49	75.0%
	Yes	93	698	88.2%
	Overall Percent	23.8%	76.2%	85.7%

Dependent Variable: If respondent experiences diminishing interest in Social Media?

4.2.4 Absence Indicators

Absence indicators are measurable quantities within the systems that would show a change in behavior of a user, for our model we investigate those quantities that will indicate the user is either passive or has totally not appeared. We obtain possible indicators from the user's activities within the systems and study their behavior after a given period of time; by quantifying these indicators we can learn their weights and how they contribute to a user's presence in the system. Table 5 shows a descriptive statistic of these factors obtained from the online questionnaire.

These behavioral statistics indicators represent a social media user's behavior online; most social media sites have one form or another of these quantities. Facebook maps directly to each and every measure while in twitter, we need to redefine the parameters to fit the model of communication. More on tweet parameters is discussed in the next chapter as per the model description.

Table 5: Descriptive statistics of the absence indicators

Descriptive Statistics

	Mean	Std. Deviation	N
Average Daily Social Media Time	2.66	1.102	989
Frequency of Chat with Friends	3.65	1.444	989
Frequency of Posting Comments	2.96	1.323	989
Frequency of Status Updating	2.47	1.203	989
Frequency of Uploading Images	1.68	1.031	989
Frequency of Uploading Videos	1.44	.996	989
Frequency of Downloading Items	2.38	1.440	989
Frequency of Viewing Other People's Profiles	3.00	1.478	989
Frequency of Reposting Other Status	2.17	1.293	989
Average Daily Data Downloads	2.84	1.228	876
Frequency of Updating Social Media Status	1.51	.725	966

Table 7 shows the correlation between these factors and the Temporal Absence. From this, we observe the strong correlation factors to be (and in considering this our dependent variable is the daily social media time which is a reverse reflection of the average social media absence period):

- Frequency of Updating Status with a Pearson Correlation of 0.478, which is the strongest correlation achieved from our data. this strong correlation shows that the more frequent a user updates their status, the more they will continue being online, On Twitter, this factor can be deduced as the number of tweets, or the frequency thereof.
- Frequency of Chat with Friends with a Pearson Correlation of 0.450 which would be interrelated to mean that the more a user chats with fellow users on social media, the more the probability they will be present in the system. For Twitter, this factor would be substituted for number of mentions on the tweets. It could not be translated to Direct

Messages since these are used majorly for privacy purposes on Twitter and the values here with, do not come in the data sets sent by twitter. It is therefore difficult to use direct messages as one of the factors.

- Frequency of Reposting Other Status with a Pearson Correlation 0.403; This gave out very strong correlation even though in like Facebook, this is not a direct mapping and would be translated into Facebook's *post sharing model*. It is a two faced factor in that the user can be more encouraged when people share their content or if they share other people's content. Twitter has a direct mapping of this factor called the *retweets* which also has two faces. For this research we define these two faces as shown in the table below for Facebook and Twitter respectively.

  Table 6: Twiter and Facebook faces of status repost

Facebook	**Twitter**
1. *Shares out*: All the reposting other users' contents done by a user	*Retweets out*: All tweets for a user that have been retweeted from other users
2. *Shares in* : All the reposting by other users of a user's contents	*Retweets in*: All tweets by a user that has been retweeted by other users.

- Frequency of Posting Comments which has a Pearson Correlation 0.345. Comments on posts by other users provide another factor that can indicate whether a user is active or absent. Considering Facebook, this is a direct mapping to the friends' comments and to a higher levels the likes and dislikes buttons are also another form of comments and user feedback. The contents of the comments themselves if analyzed can provide a direction on the mood and attitude of the user and whether their comments suggest existence or absence. On Twitter, this is a bit different and we pick the *Tweet replies* as a

representation of this, there is extra complication since tweet replies are also Tweets in themselves but that is covered by the average number of Tweets factor.

- Frequency of Viewing Other People's Profiles with a Pearson correlation of 0.234. To measure this factor, there needs to be a monitor within the systems that counts requests for other people's profiles, most suited for dating sites. In Facebook, this is highly weighted since it is common to view a person's profile before befriending them. In Twitter, it becomes a minor point.
- Frequency of Uploading Images with a correlation of 0.232 is a pointer to the behavior of a user as far as sharing content is concerned; images, URLs and other contents are common in Both Facebook and Twitter. A measure of this value gives a picture of not only a user's resourcefulness on social media but also the amount of time they possess to actually share their content.
- Other factors that are not so strong and may have less weight in prediction include: Frequency of Downloading Items with a correlation of 0.165, Average daily data downloads with a correlation of 0.137 and Frequency of Uploading Videos with a correlation of 0.023.

Tables 7a and 7b shows the correlation of these factors with the Absence duration of a user.

Table 7a: Absence Indicators Correlations I

Correlations		Average Daily Social Media Time	Frequency of Chat with Friends	Frequency of Posting Comments	Frequency of Updating Status	Frequency of Uploading Images
Average Daily Social Media Time	Pearson Correlation	1	.450**	.344**	.478**	.232*
	Sig. (2-tailed)		.000	.001	.000	.022
	Sum of Squares and Cross-products	117.888	69.551	48.653	61.490	25.561
	Covariance	1.215	.717	.502	.634	.264
	N	989	989	989	989	989

**. Correlation is significant at the 0.01 level (2-tailed). *. Correlation is significant at the 0.05 level (2-tailed).

Table 7b: Absence Indicators Correlations II

**. Correlation is significant at the 0.01 level (2-tailed). *. Correlation is significant at the 0.05 level (2-tailed).

Correlations		Frequency of Downloading Items	Frequency of Viewing Other People's Profiles	Frequency of Reposting Other Status	Average Daily Data Downloads	Frequency of Updating Social Media Status
Average Daily Social Media Time	Pearson Correlation	.165	.234*	.403**	.137	.242*
	Sig. (2-tailed)	.104	.020	.000	.205	.018
	Sum of Squares and Cross-products	25.459	37.000	55.724	16.333	18.312
	Covariance	.262	.381	.574	.190	.193
	N	989	989	989	876	966

4.3 Conclusion

We have been able to establish the existence of temporal absence and its significance in recommendations especially from our analysis of the Netflix prize data which in itself had already had early indications of strong importance during the KDD Cup completion (Takács G.

et al, 2007 and KorenY., 2008). We have then analyzed social media temporal absence data from which we have determined factors and aspect of absence, fitting them into multilayer perceptron neural network models and evaluating their behavior. From these factors, we now fit a Twitter temporal absence model using linear regression with gradient descent algorithms in the next chapter. It is otherwise worth noting that the applicability of these factors in prediction is not limited to the model we are going to discuss in the following chapter but a plethora of applicability and modeling domains exist.

CHAPTER 5

TWITTER TEMPORAL ABSENCE MODEL

The factors discussed in the previous chapters do not directly map on twitter social network; in this thesis, we match ten twitter behavioral factors to those discussed before as shown in table 8.

Table 8: Twitter Representation of Absence Factors

Absence Factors	**Twitter Representation**
Average Daily Social Media Time	Tweet Rate / Absence Index
Frequency of Chat with Friends	Number of Mentions (mentions-in, and mentions-out)
Frequency of Posting Comments	Replies
Frequency of Status Updating	Tweet Counts
Frequency of Uploading Images	Symbols, Urls, Hashtags
Frequency of Uploading Videos	
Frequency of Downloading Items	
Frequency of Viewing Other People's Profiles	--------
Frequency of Reposting Other Status	Retweets (Retweets-in, and Retweets-out)
Average Daily Data Downloads	--------
Frequency of Updating Social Media Status	--------

From table 8, there is no twitter representation of some temporal absence factors including frequency of viewing other people's profiles, average daily data downloads and frequency of updating social media status. This is due to the model of twitter as a social network site, (URL-4, 2014) details these differences, stating that "Twitter keeps people updated on the here and now, and topics and trending conversations are constantly changing. It's less about social connections and more about staying informed." Due to the constraints of the model that separates it from friendship based social media networks like Facebook, it has little relation to the three factors.

From the comments in (URL-4), the following are fundamental differences between the two social media models:

- Lifespan of content: Statistically, it has been reported that 92% of all activity and engagement with Tweets happens within the first hour of the post being made. With Facebook status updates, the interaction can go on for hours, and even days. Twitter is less about social friendships. People make connections on Facebook with friends, family members, and other people that they care to keep in touch with. Twitter allows people to follow important topics, people, and conversations that are relevant or interesting to them. It's a much more detached connection.
- Timely matters versus timelessness: While content on Facebook is timeless, the information on Twitter is always here and now, important and relevant, and constantly changing. If you have instant news, Twitter is the place to go. For content that is less focused on time or timing, Facebook is a great medium.
- Optimal times for posting are different. It has been studied and 5:00 P.M. seems to be the optimal time for tweeting, while noon seems to be the time when Facebook status updates get the most attention. Also, Wednesdays are more popular for tweets while Saturdays are more popular for Facebook posting. This is an obvious point of the difference in relevant news and content on Twitter versus the entertainment factor that Facebook has.

5.1 Tweet Factors

The absence index, is an average computation of the time differences between the end of a user's active period and the start of his/her next active time period the cumulative average of these values would form a bias for computation of a prediction as a reference point. Even though the absence index can show an indication of the frequency of a user's tweets and how active a user is online, the Tweet rate is a more accurate measure; in this thesis, the tweet rate is computation for the mean of the time differences between two tweets during the a user's active period. Below is a demonstration of these two values as modeled in this study.

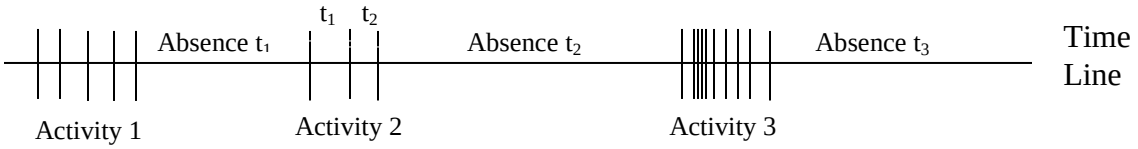


Figure 11: Absence values on Twitter activity timeline

$$Absence\_index = \left[\frac{Absence\ t_1 + Absence\ t_2}{2}\right] + Absence\ t_3\ /2$$

Tweet rate for Activity period 2 would be obtained by

$$\text{Tweet\_rate} = \left[\frac{t_1 + t_2}{2}\right]$$

The number of mentions indicates the quality of a person's tweets or otherwise their attitude towards other people's tweets; *mentions-in* refers to the total number of tweets by other users in which the user under study has been mentioned. This figure is an indication of how other users rate the current user's tweet or how much they appreciate him/her. From this research, the higher

this figure is, the most likely the user to tweet more. If a user feels appreciated and that their thoughts impact people, they will stay more online and to tweet more.

From this study we noticed that the opposite of this measure, *mentions-out* which refers to the number of tweets by the current user that mention others has an equal impact on a user's activity online; the higher the number of tweets in which a user mentions others is an indication of the degree to which the user enjoys the contents of others and translates to their readiness to participate in their social space.

The other two factors *retweets-in* and *retweets-out* have the same indication of a user's appreciation of the online space except that with these latter factors there is an inherent exposition that the user either has no time to respond to the tweets or to think of their own ideas but on the other hand, it's a measure of their excitement online. A user that changes their behavior online from mentions and replies to retweets is most probably not going to be present for longer.

5.2 Creating the Recommendation Matrix

To obtain these values, an algorithm is employed that uses the Twitter API to log in and mine tweets, analyze them while separating the negative from the positive tweets and builds a matrix (shown in table 9).

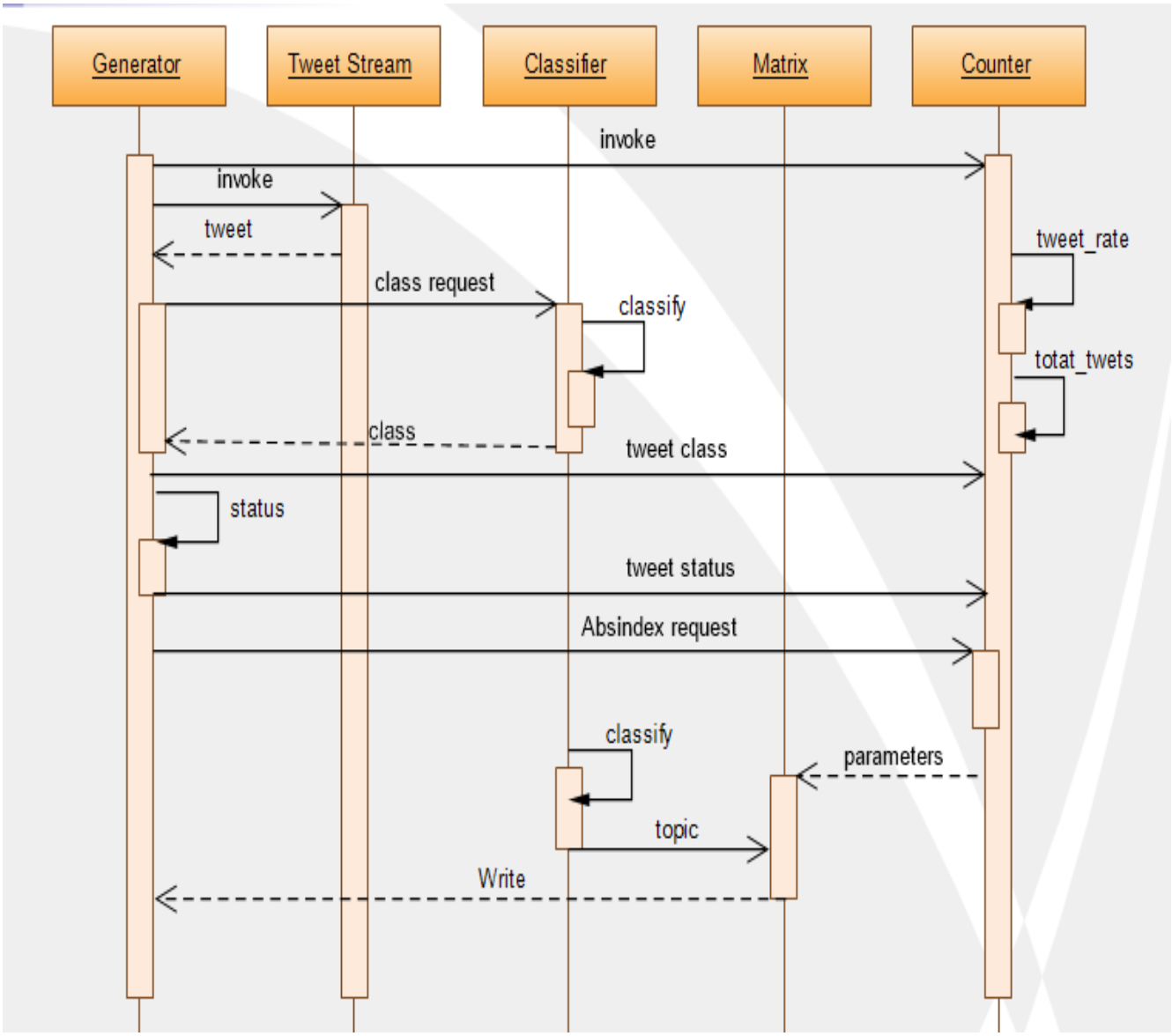


Figure 12: Sequence Diagram for the Matrix Builder Algorithm

Algorithm 3: Factors Matrix Builder Algorithm

INPUTS:
P: A decaturple of parameter values before noticed absence
A: A decaturple of parameter values after noticed absence
total_tweets ←0, pos←0, neg ←0, curr ←0, i←0
status: mention←0,
t_0 ← time stamp at first tweet
tweet_rate := 3600
$abs_{index} \leftarrow 0$
OUTPUTS: P, A, abs_{index}

Prediction_Factors()
Repeat
i := i+1
$curr := t_i - t_{i-1}$
if greater $(curr > 2 * tweet\_rate)$
$a_{(i-1,\ 2\ldots)}$
:=$\{curr, pos, neg, uploads, retweets_{out}, retweets_{in}, mentions, topic,$ total_tweets $\}$

If $abs_{index} = 0$ then
$abs_{index} = curr$
else
$abs_{index} := \frac{(abs_{index} + \text{curr})}{2}$
p_i:=
t_{i-1},
$curr, pos, neg, uploads, retweets_{out}, retweets_{in}, mentions, topic,$ total_tweets $\{\}$
$a_{(i,1)} := t_i$
reset ($pos, neg, uploads, retweets_{out}, retweets_{in}, mentions, topic,$ total_tweets$\}$
else
total_tweets := total_tweets+1
update retweet status:
if isTweet(outRetweet) then
$retweets_{out} := retweets_{out} + 1$
else if isTweet(inRetweet)
$retweets_{in} := retweets_{in} + 1$
else if isTweet(mention)
$menstions := mentions + 1$
$tweet\_rate := \frac{(tweet\_rate + \text{curr})}{2}$
tweet_class(*tweet*)? pos:= pos+1, neg:= neg+1
Until till i_n

:: This algorithm builds two matrices; pre-absence factors matrix and post absence factors matrix P & A and produces an absence index Abs_{index} these two matrices are to be used in a modeling algorithms [we divide them into training and testing sets]. In this thesis we apply a linear regression analysis model.

When algorithm 3 is run, it first logs into Twitter with the twitter handle to be used, and then mining engine fetches tweets and passes them to the classifier which uses a naïve Bayesian algorithm, to determine whether they are either positive or negative tweets. The counter module counts the number of tweets that contain given parameters; these are then used to build a matrix such as the one shown in table 9 in which the first column indicate the timestamp at the start of the absence period and the factor values are given of the values collected .

Table 9: Twitter Temporal Absence factors matrix

Timestamp	positive	negative	hashtags	media	urls	symbols	retweets	mentions	total_tweets
1/15/2013 16:16	3	3	2	0	4	0	4	3	6
1/15/2013 21:26	0	2	1	0	2	0	1	1	2
1/16/2013 16:51	0	1	0	0	0	0	1	2	1
1/17/2013 15:48	4	15	1	0	3	0	2	2	19
1/18/2013 15:07	5	2	3	5	4	0	3	6	7
1/19/2013 14:01	5	2	2	3	5	0	3	6	7
1/20/2013 13:46	7	3	2	4	4	0	7	5	10
1/21/2013 13:32	6	3	2	1	4	0	4	8	9

5.3 Regression Model

This thesis proposes a regression model hypothesis

$$h(x) = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \cdots + \theta_n x_n$$

Where x are the values of the factors obtained from the matrix and $\theta$is the weight for the respective parameter.

$$h(x) = \sum_{i=0}^{n} \theta_i \, x_i = \ \theta^T X$$

Given Training sets P/A, we fit features X and actual results Y, the task is to make the difference between prediction and the actual values minimal. A cost function using the mean square errors is therefore thus defined:

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{n} (h_\theta x^{(i)} - y^{(i)})^2$$

Gradient Descent update θ as follows

$$\theta_i \leftarrow \theta_i - \ \alpha \frac{\delta}{\delta \theta_i} J(\theta)$$

$$\frac{\delta}{\delta \theta_i} J(\theta) = \frac{\delta}{\delta \theta_i} \frac{1}{2} (h_\theta(x) - y)^2$$

Applying chain Rule

$$= 2 * \frac{1}{2} \ (h_\theta(x) - y) * \frac{\delta}{\delta \theta_i} (h_\theta(x) - y)$$

$$= (h_\theta(x) - y) * \frac{\delta}{\delta \theta_i} (h_\theta(x) - y)$$

$$= (h_\theta(x) - y) * x_i$$

Partial differentiation with respect to θ_i where only one term has θ_i; the term $\theta_i x_i$

$\therefore \theta_i \leftarrow \theta_i - \alpha(h_\theta(x) - y) * x_i$

Where α is the learning rate of the gradient descent algorithm?

5.4 Choosing the Learning Rate

Since the learning rate α is a handpicked variable, we experimented with several values and it was found that for these parameters, an alpha in the range 0.001 to 0.08 would reach convergence, table 10 shows the θ_i values for different alphas used while figure 12 shows the graphs plotted with these alpha values for 100 iterations of the gradient descent algorithm.

Table 10: Theta Gradient Descent Values for Different Alphas

	$\alpha = 0.003$	$\alpha = 0.006$	$\alpha = 0.009$	$\alpha = 0.011$	$\alpha = 0.014$	$\alpha = 0.0155$
θ_1	2.6646528	4.0389143	4.7973880	5.1163557	5.4378997	5.5498605
θ_2	-2.3816318	-3.9049300	-4.9239659	-5.4381920	-6.0555695	-6.3200448
θ_3	-1.3920391	-2.0222269	-2.2443150	-2.2777489	-2.2425465	-2.2090560
θ_4	-1.2668285	-1.8486489	-2.0638266	-2.1024661	-2.0816321	-2.0554291
θ_5	-0.7601313	-0.9812847	-0.9493766	-0.8713663	-0.7300990	-0.6628637
θ_6	-0.9222302	-1.2450143	-1.2593835	-1.1877625	-1.0287469	-0.9434516
θ_7	-1.9738058	-3.2035968	-3.9668106	-4.3145841	-4.6798448	-4.8159546
θ_8	1.5073355	1.9812410	2.1074895	2.1091060	2.0627871	2.0194544

From figure 12 we realize that the gradient descent minimization algorithm for the cost function J(θ) converges (reaches a minimum) as fast as possible when the value of alpha is 0.014 as indicated by the doted blue curve. Also it is visible that the graph when alpha is 0.0155, indicated by dotted red curve, starts to miss the minimum and converges a little slower than expected,

figure 13 shows the graph when alpha is 0.016 which indicates that gradient descent does not converge and that J(θ) blows up into a negative graph.

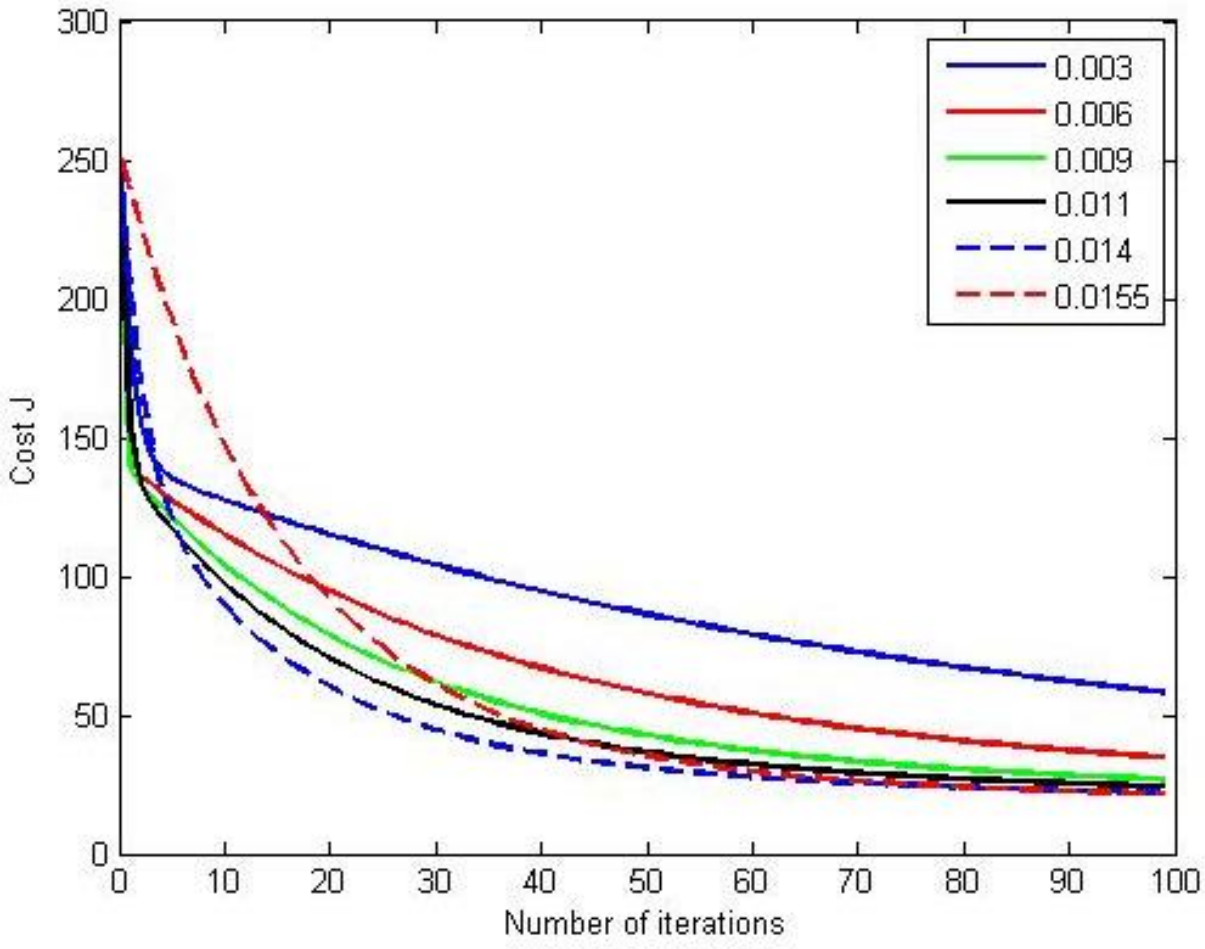


Figure 13: Plot of Cost Function $J(\theta)$ for different learning rates

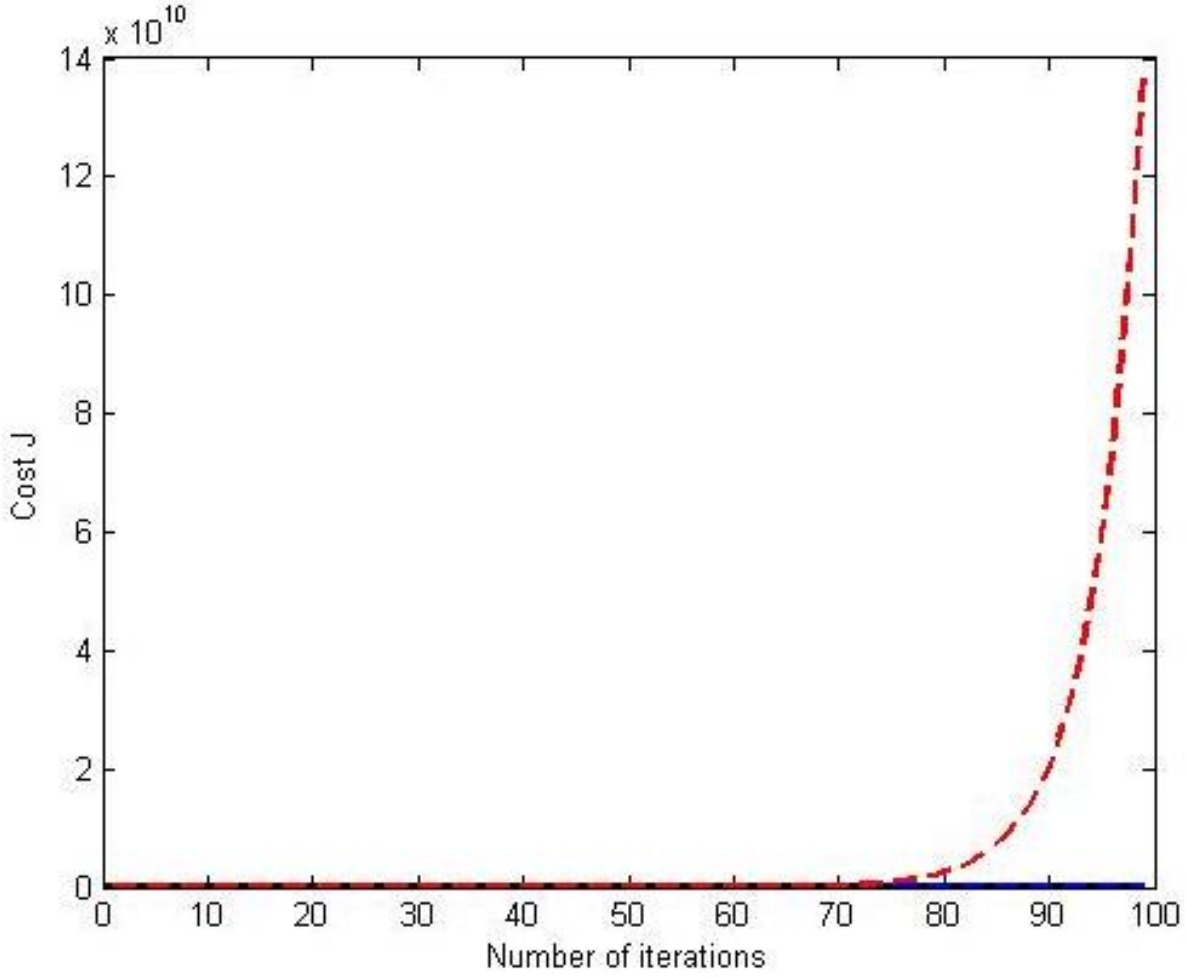


Figure 14: Plot of Cost Function $J(\theta)$ for a learning rate 0.0166

5.5 Prediction of Absence

From the findings, theta values obtained when alpha was tested at 0.014, we fit the weights into our regression hypothesis to obtain the predicted absence periods as follows.

$$h(x') = 1 + 5.4378997x_1 - 6.0555695x_2 - 2.2425465x_3 - 2.0816321x_4 - 0.7300990x_5 - 1.0287469x_6 - 4.6798448\,x_7$$

Appendix I gives the python code used Classification of tweets from online Tweeter Accounts and the implementation of the Matrix builder Algorithm can be found in Appendix II. Given below is a Matlab output for prediction of Absence given this model.

Test Data:

X1= 8 X2=6 X3=6 X4=2 X5=13 X6=4 X7=3 X8=14 **Y=19.38 hrs**

theta_grad_descent =

5.437899690785792

-6.055569456977179

-2.242546542479401

-2.081632082000593

-0.730098957670007

-1.028746857399485

-4.679844802794265

2.062787170265663

1.710180792955578

Absence_grad_desc = -20.989504178985086

theta_normal =

16.460695732178252

-3.019790380666766

-0.035270326387331

-0.336851196048456

0.199660855472963

0.564290096956136

-0.211593301607079

-0.493630265808756

1.520041381852871

Absence_normal =

16.758051865690710

The Value 16.758 is a value of 86.5 Accuracy

The absence value is generated at an accuracy level of 86.5% with this linear regression model. From the results, the $\boldsymbol{\theta_i}$ (i.e. the weight of each factor: $\boldsymbol{x_i}$) value displayed for the theta_grad_descent are the factor values obtained by the gradient function before minima is reached while the theta_normal values show the weights of each $\boldsymbol{\theta_i}$ factor after the minima is reached.

CHAPTER 6

CONCLUSION AND FURTHER WORK

This chapter summarizes the work presented in this thesis and discusses further work to express possible modeling directions or application domains for predicting temporal absence or utilizing the same as sub models for other applications.

6.1 Summary of the Thesis

This thesis provides a working definition of Temporal Absence in the context of Recommendation Systems based on the temporal behavior of users. There is subsequently a study of existing recommendation systems through the KDD Cup Netflix prize data which asserted the existence of temporal absence within these recommendation systems; after this determination, this study sought to determine the factors that influence or affect the absence of users in Social Media and the overall users' satisfaction with current social media recommendations and their expectation. From the online questionnaire that gave 989 analyzed responses, both factors and aspects of temporal absence were determined, these might not be the only factors or aspects as far as temporal absence or social media usage is concerned but they were recognizable from the data and sufficient for the modeling thus proposed.

In this thesis, a model has also been discussed, that employs a linear regression hypothesis trained by gradient descent algorithm that predicts absence duration to an accuracy of 86.5%. The data used in this model are live tweets obtained from twitter's API. This data was then divided into training sets and probe set. The model is trained using the batch gradient descent algorithm and an appropriate alpha is picked to determine the weights (**thetas**).

From this research, therefore, we can conclude that the temporal absence is inherent within recommendation systems and Social Media and that the factors that lead to Absence can be fitted in machine learning algorithms to predict a user's absence duration.

6.2 Future Work

The specific application of the absence duration so obtained is not within the scope of this research but this work proposes that this information would be vital in predicting traffic within systems, in chat based applications; absence predictions would assist in helping users avoid waiting on other users who have fallen absent. Other Application domains are in the Matrix Factorization for E commerce Recommendations; this research fosters a notion an absent user has a high probability of changing preferences within the system hence his ratings on items would vary from the normal. Calculating this probability and subsequently applying the same in ratings prediction is an area for further research.

A Linear Regression model with gradient descent has been applied in this research as the prediction algorithm but that does not limit the use of other algorithms like the Neural Networks or Support Vector Machines for the same purpose, it is suggested that the modeling could be better if other algorithms were employed but this would require an expanded data mined from social media and other Linked Open Data datasets.

During the modeling process, the absence recommendation matrix derived from Twitter appeared to be near singular matrix which is one of the factors influencing the error value; this shows that there is need for use of more factors or otherwise preparation of the matrix before training. An area of further research could be on how to improve the absence matrix to get more accurate absence duration recommendations.

CHAPTER 7

REFERENCES

1. Alexandros Karatzoglou and Markus Weimer, "*Quantile Matrix Factorization for Collaborative Filtering*", 2010
2. Bell R., Volinsky C. "*Matrix factorization techniques for recommender systems*". AT&T Labs—Research, August 2009.
3. Bell Robert M. and Yehuda Koren. "*Improved Neighborhood-based Collaborative Filtering*". *KDDCup'07,* August 12, 2007, San Jose, California, USA.
4. Billsus, D., & Pazzani, M. J., "*Learning collaborative information filters*". In Proceedings of the fifteenth international conference on machine learning (ICML – 98), Madison, Wisconsin. San Francisco: Morgan Kaufmann 1998.
5. Castillejo E., Almeida A., and L´opez-de-Ipi ˜na D., "Alleviating cold-user start problem with users' social network data in recommendation systems". 2012. Deusto Institute of Technology - DeustoTech, University of Deusto, Avda. Universidades 24, 48007 - Bilbao, Spain.
6. Frias-Martinez, E., Magoulas, G., Chen, S. Y., and Macredie, R., "*Automated user modeling for personalized digital libraries*", International Journal of Information Management, Vol. 26, 2006, pp.234 –248.
7. Frias-Martinez, E., Chen, S. Y., and Liu, X. "*Evaluation of a personalized digital library based on cognitive styles: Adaptivity vs. Adaptability*", International Journal of Information Management, Vol. 49, 2009, pp.48–56
8. Herlocker J.L., Konstan J.A., Borchers A., and Riedl J. "*An algorithmic framework for performing collaborative filtering*". In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, page 237. ACM, 1999

9. Herna ´ ndez F., Gaudioso E., "*Evaluation of recommender systems: A new approach*". Elsevier Ltd, 2007.

10. Http://s1.zetaboards.com/RPG_Outcry/topic/3034446/1/ accessed on June 21, 2012.

11. Jahrer M., Töscher A., Legenstein R., "Combining predictions for accurate recommender systems", Proceedings of the 16th ACM RecSys Conference, 2010.

12. Kirk Baker., "*Singular Value Decomposition Tutorial*". March, 2005.

13. Koren Yehuda. "*Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model"*. *KDD'08,* August 24–27, 2008, Las Vegas, Nevada, USA, 2008.

14. Koren Yehuda. "Collaborative Filtering with Temporal Dynamics", KDD'09, June 28–July 1, 2009, Paris, France. Copyright 2009 ACM 978-1-60558-495-9/09/0

15. Koren Y., and Bell R., "Advances in Collaborative Filtering", Springer Science + Business Media, LLC, 2011.

16. Lee H.J., Kim J.W., Park S.J., "*Understanding collaborative filtering parameters for personalized recommendations in e-commerce*", Springer, 2007.

17. Lathia N., Hailes S., Capra L. and Amatriain X., "Temporal Diversity in Recommender Systems". SIGIR'10, July 17-23, 2010, Geneva, Switzerland. ACM 978-1-60558-896-4/10/07

18. Li Chen and Pearl Pu, "User Evaluation Framework of Recommender Systems", Workshop SRS '10, February 7, 2010, Hong Kong, China. Copyright, 2010 ACM 978-1-60558-995-4/10/01.

19. Li Chen and Pearl Pu, "Evaluating recommender systems from the user's perspective: survey of the state of the art". Springer Science +Business Media B.V. March, 2012.

20. Long Jin, Yang Chen, Tianyi Wng, Pan Hui, Athanasios Vasilakos., "Understanding User Behavior in Online Social Networks: A Survey". IEEE Communications Magazine, September, 2013.

21. Mark Graus., "Understanding the Latent Features of Matrix Factorization Algorithms in Movie Recommender Systems "Eindhoven, March 25th, 2011.

22. Melville P., Sindhwani V., "*Recommender Systems*". Encyclopedia of Machine Learning, Springer-Verlag Berlin Heidelberg, 2010.

23. Moreno A., Ariza-Porras C., Lago P., Jimenez-Guarin C., Castro H., and Riveill M., "Hybrid model rating prediction with Linked Open Data for Recommender Systems". ESWC 2014 Linked Open Data-enabled Recommender Systems Challenge Task 1

24. Ngoc – Diep Ho, *"Nonnegative matrix factorization algorithms and applications",* June 2008.

25. Park S., and Chu W.. Pairwise Preference Regression for Cold-start Recommendation. RecSys'09, October 23–25, 2009, New York, New York, USA. Copyright 2009 ACM 978-1-60558-435-5/09/10

26. Paterek A., "*Improving Regularized Singular Value De -composition for Collaborative Filtering*," Proc. KDD Cup and Workshop, ACM Press, 2007, pp. 39-42.

27. P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, Grouplens: "*An open architecture for collaborative filtering of netnews"*, in Proceedings of CSCW, ACM SIG Computer Supported Cooperative Work, 1994.

28. Purushotham S., Liu Y., Kuo J., "*Collaborative Topic Regression with Social Matrix Factorization for Recommendation Systems*" in Proceedings of the 29th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012.

29. Rashid A.M., Karypis G., Riedl J., "*Influence in Ratings-Based Recommender Systems: An Algorithm-Independent Approach"*. 2005.

30. Sarwar, B., Karypis, G., Konstan, J., & Reidl, J. *"Item – based collaborative filtering recommendation algorithms".* In WWW '01: Proceedings of the tenth international conference on World Wide Web (pp. 285–290). New York: ACM. Hong Kong,

31. Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M.*Methods and metrics for cold-start recommendations*". In Proceedings of the ACM SIGIR, 2002, Tampere, Finland (pp. 253–260).

32. Slobodan V., Zoran O., "*Collaborative Filtering Using a Regression-Based Approach*", Springer-Verlag London Ltd, 2004.

33. Srebro N. and Jaakkola T. "*Weighted low – rank approximations*". In Proc. of ICM L-03, 20th Int. Conf. on Machine Learning, p ages 720–727, Melbourne, Florida, US A, 2003.
34. Srebro, N., Rennie, J., Jaakkola, T.: Maximum-margin matrix factorization. In Saul, L.K., Weiss, Y., Bottou, L., eds.: Advances in Neural Information Processing Systems 17 NIPS, Cambridge, MA, MIT Press, 2005.
35. Takács G., Pilászy I., Németh B., and Tikk D., "On the Gravity recommendation system. In Proc. of KDD Cup Workshop at SIGKDD - 07, 13th ACM Int. Conf. on Knowledge Discovery and Data Mining, pages 22–30, San Jose, California, US A, 2007.
36. Terveen L., and Hill W., "*Beyond Recommender Systems: Helping People Help Each Other*". AT&T Labs - Research; In HCI In The New Millennium, Jack Carroll, ed., Addison-Wesley, 2001.
37. Takács G., Pilászy I., and Németh B., *"Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem", ACM, 2008*
38. URL-1:www.goal.com/en-us/news/3869/fifa-ballon-dor/2014/01/14/4543402/the-making-of-cristiano-ronaldo-how-the-2013-ballon-dor, Jan, 14, 2014.
39. URL-2: Http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/., 12th, July 2012, Accessed Nov, 28, 2012.
40. URL-3: Simon Funk, "*Netflix Update: Try This at Home,*" Dec. 2006; http://sifter.org/~simon/journal/20061211.html. Accessed November, 22, 20012.
41. URL-4: http://www.visualscope.com/twitfb.html., Accessed Sep, 29, 2014.

Appendix I: Tweet Classifier Code

```
import csv

def get_training_data():

    '''Get the training data from the the input file.

    Return a list of [[label,tweet]] for each tweet in the training set.

    The tweet part is a list of words in the tweet.

    e.g. ['positive', ['happy', 'birthday', 'the', 'fantabulously', '@shrimponbarbie']]

    '''

    f=open('train.csv','rb')  #training file

    reader=csv.reader(f,delimiter=',')

    training_data = []  #list that will hold the training tweets

    for row in reader:  #loop through the entire file

        training_data.append(row)   # add each label, tweet pair to the list


    training_data=[item for item in training_data if len(item)==2] #ensure that each item in the set is of length 2


    for i in range(len(training_data)-1): #loop through entire training set

        if not isinstance(training_data[i][1],list): #check whether tweet part of training set is a list

        #turn the tweet into a list of words in lower case for words that have a length of at least 3

        #lower case words makes comparison easier so that 'good' and 'Good' are taken as the same word

        #checking word length ensures that words such as 'a', 'an', 'is', 'be' etc. are eliminated from

        #the pool of words and may not add any value

            training_data[i][1]=[word.lower() for word in training_data[i][1].split()]
```

```
    f.close()

    return training_data

def get_test_tweets(tweets):

    test=[]

    for tweet in tweets:
        test.append(['',tweet['text'].encode('utf-8')])

    for i in range(len(test)-1): #loop through entire training set
        if not isinstance(test[i][1],list): #check whether tweet part of training set is a list
        #turn the tweet into a list of words in lower case
                test[i][1]=[word.lower() for word in test[i][1].split()]
    return test
def get_words(data):
    '''
    Get the words in the training/test set.
    Returns a list of the unique words(vocabulary) in the data
    '''
    words = []  #list to hold all the words in the
    for item in data:
        words.extend(item[1])
```

```
    return list(set(words))


# Get Probability of each word in the training data
# If label is specified, find the probability of each word in the corresponding labelled tweets only
def get_word_prob(training_data, label = None):
    '''
    Get the probability of each word in the training/test data.
    The label is optional.
    Returns a dictionary with key, value pairs of word and probability
    '''
    words = get_words(training_data)    #list of all unique words in the set
    freq = {}   #dictionary to store frequency of each word in the set

    for word in words:
        freq[word] = 1  #initialize count of each unique word to 1

    total_count = 0     #variable to store the count of all words
    for data in training_data:
        if data[0] == label or label == None:
            total_count += len(data[1]) #increment word count for each word
            for word in data[1]:    #loop through every word in tweet part of training data
                freq[word] += 1     #increment count of each word in tweet for each tweet

    prob = {}   #dictionary to store the probability of each word in the set
    for word in freq.keys():
```

```
        prob[word] = freq[word]*1.0/total_count #calculate probability of each word


    return prob


def get_label_prob(training_data, label):
    '''Get the probability for a given label
    Returns the probability value for each label as per the training set
    '''
    label_count = 0
    total_count = 0
    for data in training_data:
        total_count += 1    #increment total stepwise
        if data[0] == label:
            label_count += 1    #increment label count stepwise
    return label_count*1.0/total_count


# Label the test data given the trained parameters Using Naive Bayes Model
def label_data(test_data, positive_word_prob, negative_word_prob, positive_prob, negative_prob):
    labels = []
    count={}
    count['positive']=0
    count['negative']=0
    for data in test_data:
        data_prob_positive = positive_prob
```

```
    data_prob_negative = negative_prob

    for word in data[1]:
        if word in positive_word_prob:
            data_prob_positive *= positive_word_prob[word]
            data_prob_negative *= negative_word_prob[word]
        else:
            continue

    if data_prob_positive >= data_prob_negative:
        labels.append([' '.join(data[1]), 'positive', data_prob_positive, data_prob_negative])
        count['positive']+=1
    else:
        labels.append([' '.join(data[1]), 'negative', data_prob_positive, data_prob_negative])
        count['negative']+=1
pos='pos: '+str(count['positive'])
neg='neg: '+str(count['negative'])
nulist=[count['positive'],count['negative']]
return nulist.
```

Appendix II: Matrix Generation Code

```
from login import *
from classifier import *
from urllib2 import URLError
from  httplib import BadStatusLine
import twitter
import json
import sys
from sys import maxint
from functools import partial
from datetime import datetime, timedelta

import pandas as pd
import csv

def make_twitter_request(twitter_api_func, max_errors=3, *args, **kw):
    # A nested helper function that handles common HTTPErrors. Return an updated
    # value for wait_period if the problem is a 503 error. Block until the rate
    # limit is reset if a rate limiting issue
    def handle_twitter_http_error(e, wait_period=2, sleep_when_rate_limited=True):

        if wait_period > 3600: # Seconds
            print >> sys.stderr, 'Too many retries. Quitting.'
            raise e
```

```
# See https://dev.twitter.com/docs/error-codes-responses for common codes


if e.e.code == 401:
    print >> sys.stderr, 'Encountered 401 Error (Not Authorized)'
    return None
elif e.e.code == 404:
    print >> sys.stderr, 'Encountered 404 Error (Not Found)'
    return None
elif e.e.code == 429:
    print >> sys.stderr, 'Encountered 429 Error (Rate Limit Exceeded)'
    if sleep_when_rate_limited:
        print >> sys.stderr, "Sleeping for 15 minutes...ZzZ..."
        sys.stderr.flush()
        time.sleep(60*15 + 5)
        print >> sys.stderr, '...ZzZ...Awake now and trying again.'
        return 2
    else:
        raise e # Allow user to handle the rate limiting issue
elif e.e.code in (502, 503):
    print >> sys.stderr, 'Encountered %i Error. Will retry in %i seconds' % \
        (e.e.code, wait_period)
    time.sleep(wait_period)
    wait_period *= 1.5
    return wait_period
```

```
        else:
            raise e

    # End of nested helper function

     wait_period = 2
    error_count = 0

    while True:
        try:
            return twitter_api_func(*args, **kw)
        except twitter.api.TwitterHTTPError, e:
            error_count = 0
            wait_period = handle_twitter_http_error(e, wait_period)
            if wait_period is None:
                return
        except URLError, e:
            error_count += 1
            print >> sys.stderr, "URLError encountered. Continuing."
            if error_count > max_errors:
                print >> sys.stderr, "Too many consecutive errors...bailing out."
                raise e

def harvest_user_timeline(twitter_api,screen_name=None,user_id=None, max_results=3200):
```

```
assert(screen_name != None) != (user_id != None), \
"Must have screen_name or user_id but not both"

kw={ #keyword args for the Twitter API call
'count':200,
'trim_user': 'true',
'include_rts':'true',
'since_id':1      }

if screen_name:
        kw['screen_name']=screen_name
else:
        kw['user_id']=user_id

max_pages=16
results=[]
tweets=make_twitter_request(twitter_api.statuses.user_timeline,**kw)
if tweets is None:
        tweets=[]

results+=tweets
print sys.stderr, 'Fetched %i tweets' %len(tweets)
page_num=1

if max_results == kw['count']:
```

```
            page_num=max_pages
        while page_num < max_pages and len(tweets) >0 and len(results) <max_results:
            kw['max_id']=min([tweet['id'] for tweet in tweets])-1

            tweets=make_twitter_request(twitter_api.statuses.user_timeline, **kw)
            results+=tweets
            print sys.stderr, 'Fetched %i tweets' %(len(tweets),)
            page_num+=1

        print sys.stderr, 'Done fetching tweets'
        return results[:max_results]
def strip(date):
    return datetime.strptime(date,'%a %b %d %H:%M:%S +0000 %Y')
def get_user_retweets(tweets):
    user_retweets=0
    for tweet in tweets:
        if tweet.has_key('retweeted_status'):
            user_retweets+=1
    return user_retweets

def get_user_mentions(tweets):
    screen_names=[user_mention['screen_name']
                for status in tweets
                    for user_mention in status['entities']['user_mentions']]
```

```
    return len(set(screen_names))
    def get_hashtags(tweets):
    hashtags=[hashtag['text']
                for tweet in tweets
                    for hashtag in tweet['entities']['hashtags']]

    return len(set(hashtags))
def get_urls(tweets):
    urls=[url['expanded_url']
            for tweet in tweets
                for url in tweet['entities']['urls']]

    return len(set(urls))

def get_symbols(tweets):
    symbols=[symbol['text']
                for tweet in tweets
                    for symbol in tweet['entities']['symbols']]

    return len(set(symbols))

def get_media(tweets):
    media=[]
    for tweet in tweets:
        if tweet['entities'].has_key('media'):
```

```
            media.append(tweet['entities']['media'][0]['url'])
        return len(set(media))
def create_matrix(tweets):
    hashtags=get_hashtags(tweets)
    media=get_media(tweets)
    urls=get_urls(tweets)
    symbols=get_symbols(tweets)
    mentions=get_user_mentions(tweets)
    retweets=get_user_retweets(tweets)
    total_tweets=len(tweets)
    test_data = get_test_tweets(tweets)
    test_labels = label_data(test_data, positive_word_prob, negative_word_prob, positive_prob,
negative_prob)
    #print test_labels
    pos=test_labels[0]
    neg=test_labels[1]
    matrix=[pos,neg,hashtags,media,urls,symbols,retweets,mentions,total_tweets]


    return matrix
def create_dataframe(init,fin,labels,timestamp):
     df=pd.DataFrame(data={'init':init,'fin':fin},columns=['init','fin'],index=labels)
     df.columns.name=timestamp
     return df
#def get_matrix(tweets):
```

```
def get_avg(tweets):
    times=[]
    for i in range(1,len(tweets)):
        time_diff=(tweets[i]['created_at']-tweets[i-1]['created_at']).total_seconds()
        times.append(time_diff)
    #print times
    avg=sum(times)/len(times)
    return avg


if __name__=="__main__":
    uname='barackobama'
    abs_index=0
    twitter_api=oauth_login()
    tweets =harvest_user_timeline(twitter_api, screen_name=uname,max_results=10000)
    tweets.reverse()
    filname=uname+'.csv'
    with open(filname,'w') as fil:
        writer=csv.writer(fil)

labels=['timestamp','positive','negative','hashtags','media','urls','symbols','retweets','mentions','tota
l_tweets','absence_index']
        writer.writerow(labels)
        #twt_dct=tweets_dict(tweets=tweets)
        for tweet in tweets:
            tweet['created_at']=strip(tweet['created_at'])
```

```
tweet_rate=get_avg(tweets)
training_data = get_training_data()

word_prob = get_word_prob(training_data)
positive_word_prob = get_word_prob(training_data, 'positive')
negative_word_prob = get_word_prob(training_data, 'negative')

# Get the probability of each label
positive_prob = get_label_prob(training_data, 'positive')
negative_prob = get_label_prob(training_data, 'negative')

# Normalise for stop words
for (word, prob) in word_prob.iteritems():
    positive_word_prob[word] /= prob
    negative_word_prob[word] /= prob
i=0
j=0
final_matrix=None
#k=0
while i < (len(tweets)-2):
    i+=1
    #print i
    time_diff=(tweets[i]['created_at']-tweets[i-1]['created_at']).total_seconds()
    #print i,time_diff, tweets[i]['created_at']
    if time_diff>tweet_rate:
```

```
if abs_index==0:
    abs_index=time_diff
    #print abs_index
else:
    abs_index=(abs_index+time_diff)/2

if final_matrix == None:
    initial_matrix=create_matrix([])
    initial_matrix.insert(0,tweets[i]['created_at'])
    initial_matrix.append(0)
    writer.writerow(initial_matrix)
    #print initial_matrix
    final_matrix=create_matrix(tweets[j:i])
    final_matrix.insert(0,tweets[i]['created_at'])
    final_matrix.append(abs_index)
    writer.writerow(final_matrix)
    #print final_matrix
    #df=create_dataframe(initial_matrix,final_matrix,labels,tweets[i]['created_at'])
    #print df
    j=i
    #print j
else:
    #print i, time_diff
    initial_matrix=final_matrix
```

```
            final_matrix=create_matrix(tweets[j:i])
            final_matrix.insert(0,tweets[i]['created_at'])
            final_matrix.append(abs_index)
            writer.writerow(final_matrix)
            #print initial_matrix
            #print i,j,initial_matrix,final_matrix,abs_index,tweet_rate
            j=i
        if abs_index==0:
            abs_index=time_diff
            #print abs_index
        else:
            abs_index=(abs_index+time_diff)/2
            #print abs_index

#else:
    #tweet_rate=(tweet_rate+time_diff)/2
    #print abs_index, tweet_rate
```

Appendix III: Survey Questionnaire

Social Media Temporal Absence Recommendation: a Collaborative Filtering Perspective

Dear Respondent, I am the undersigned, currently undertaking a Masters Degree in Software Engineering at JKUAT. My research pertains to Improving the accuracy and efficiency of recommender systems by evaluating the absence patterns of users; using social media as a case study. To this end therefore, I am conducting a survey in order to establish the attributes that contribute to absence from the system and the degree to which absence in itself can contribute to predicted future behavior and preference. The whole survey will take approximately 10 minutes of your time. Please complete all of the items. Your participation in this study is voluntary and all responses will be used only for the intended purpose of the research and will be treated in the strictest of confidence. Please direct any inquiries concerning this survey to: Mulang' Isaiah Onando Email:icesiremulacs@gmail.com Phone: +254 711 250 239

---

* Required

Part 1: Background Information

1.1 You are? *

- Male
- Female

1.2. Please indicate your age group * *

- Under 18
- 19-25
- 26-40
- 41-55
- 55+

1.3 How long is your experience in using a computer

- No experience
- Less than 1 year
- 1 - 5 years
- 6 - 10 years

- 11 - 15 years
- 15+ years

1.4 What do you mainly use your computer for?

- Work
- Entertainment
- Study
- Communication
- Hobby

Others (please specify) 

1.5 How long have you been using the internet (years)? *

- Less than 1 year
- 1 - 5 years
- 6 - 10 years
- 10+ years

1.6 How would you rate your skills as an internet user? *in terms of the ease and efficiency.

- Very Basic
- Basic
- Average
- Advanced
- Very Advanced

Part 2: Social Media Use and Experience

This section contains question regarding your use and view of the social media

2.1 Are you familiar with any Social Media Network sites? *

- Yes
- No

2.2 If YES, Do you use any of the following social networking sites? (Check appropriately)

- Facebook
- Twitter
- Google+ (G+)
- Linkedln
- Myspace
- Tagged
- DeviantArt
- Meetup

2.3 If you are NOT using social networking websites, give the reasons why? Check Wherever appropriate

- I do not want any publicity.
- They are difficult to use.
- I do not have the time.
- It is not interesting for me.
- Limited internet access/facilities.
- I do not trust the sites.
- It is expensive.
- Limited internet access/facilities.

Others (Specify) 

2.4 What is your general view on the necessity of social media? *

- No Opinion
- Don't care
- Unnecessary
- Quite necessary
- Necessary
- Very Necessary

- Absolutely Necessary

Others (Specify) 

2.5 For how long have you been using social media (years) *

- Less than 1 year
- 1 - 3 years
- 4 - 6 years
- 7 - 9 years
- 10+ years

2.7 To what level do you prefer the following social media sites? *1 - Not preferred , 5 - most preffered

	1	2	3	4	5
Facebook					
Twitter					
G+					
LinkedIn					
MySpace					
Tagged					
DeviantArt					
Meetup					

2.8 Rate the following reason(s) on how they determine your preference(s) above? *1 - Negligible reason, 5 - Major reason

	1	2	3	4	5
Easy to Find and Connect to Friends.					
Easy to Use and Understand.					
Helpful Recommendations on Items and Friends.					
Attractive Interface.					
Many of your Friends use it.					
You Heard about it First.					

Others (Specify)

2.9 How would you rate the following aspects as far as your purpose for use of social media is concerned? *Change According to 3.11

	1	2	3	4	5
Keeping in touch and catch up with friends					
Education and Research work					
Enhancing and updating personal knowledge					

	1	2	3	4	5
Business and advertisement					
Chatting					
Accessing the current news					

Kindly enlist other factors not in the grid

Part 3: Recommendation Information

This section seeks to identify factors that can be used in recommendations to improve prediction accuracy and Satisfaction.

3.1 On Average, How long do you spend on social media per day? *You can combine all you time and give a rough proximate of average daily time.

- Less than 1/2 hr
- 1/2 - 1hr
- 1hr - 2hrs
- 2 - 4hrs
- 4 - 8hrs
- 8+ hrs

3.2 In a scale of 1 to 5, how much of this time are you fully active on the social media? *Active means you have dedicated time to social media within the said period without distraction

	1	2	3	4	5	
passive						Active

3.3 How do your friends know you will be absent from social media? *

- No idea
- You leave them a message
- They just realize after a while
- You don't care

3.4 To what degree do you agree that knowing when your friend will be absent from Social Media is important *

- Strongly Disagree
- Disagree
- Neutral
- Agree
- Strongly Agree

3.5 What information would you want to know on availability of friends? *

- The times they will be online
- The duration of time they spend online
- The next time they are online
- How long they take to respond to messages
- Other:

3.6 All social networks have the "people you may know / Pages you may like" recommendations, how useful do you find these recommendations *

- Extremely useful
- Very useful
- useful
- quite useful
- not useful

3.7 How often do you encounter irrelevant recommendations *

- Always
- Often
- Sometimes
- Scarcely
- Never

3.8 Why do you think the recommendations are irrellevant to you? *check as many as appropriate

- ☐ They are people you dont know
- ☐ They are people you met too long ago hence obsolete friends
- ☐ They are people you met in a period of life that you don't want to remember
- ☐ They are people of different gender
- ☐ They are people whose social life differ from yours
- ☐ They are people of different economic or academic status
- ☐ They are people you have no mutual friends with
- ☐ Other: _____

Part 4: Absence Factors

If you do not work, you may skip questions in this section from 4.12 (i.e. 4.12 to 4.17)

4.1 What is the longest duration you have ever taken before visitng a given social media site? *_____

4.2 What factor led to the prolonged absence from the given site in 4.1 above? *

- ○ I lost interest
- ○ The topic of discussion/conversation was not pleasing
- ○ Poor health (I was sick/unwell)
- ○ I was in a remote area without internet
- ○ Found a different social media site more interesting
- ○ Was too busy at my work place
- ○ Other: _____

4.3 How often do the following factors contribute to your absence from social media? *

	Never	Seldom	Sometimes	Often	Always
Busy Schedule	○	○	○	○	○
Travelling	○	○	○	○	○
Nagging Social Media Friends	○	○	○	○	○

	Never	Seldom	Sometimes	Often	Always
Lack of Response from Friends					
Nothing to do online					
Lack of Internet Connection					
Poor Health					
Not liking existing conversations					

4.4 In a period of poor health or sickness, would your visit social media sites like facebook? *

- Cant tell
- I would never visit at all
- I would visit social media less frequently/often
- I would visit social media as usual
- I would visit social media more Frequently/often
- A would always visit social media

4.5 How often do you experience poor health or abnormal body conditions that would affect your presence in social media? *

- Daily
- Weekly
- Monthly
- Once in a while
- Can't tell

4.6 What is the primary equipment you use for accessing social media sites? *

- Desktop
- Laptop
- Tablet
- Mobile phone

4.7 If you were to change your primary equipment from desktop/laptop to a tablet/mobile phone, how would this affect your frequency on social media? *

- Tremendously increase
- Slightly increase
- No effect
- Slightly decrease
- Tremendously decrease

4.8 If you were to change your primary equipment from tablet/mobile phone to desktop/laptop, how would this affect your frequency on social media? *

- Tremendously increase
- Slightly increase
- No effect
- Slightly decrease
- Tremendously decrease

4.9 What factors would cause this change of your primary access equipment? *

- Travelling
- Poor health/sickness
- When bored with my primary equipment
- When I require privacy
- For security
- Other:

4.10 Do you experience reduced/diminishing interest in social media? *

- Yes
- No

4.11 If yes, to what degree do the following contribute to your diminishing interest?

	Extremely	Very	Moderately	Slightly	Not at all
Current topics of discussion are disinteresting					
Friends are attacking you or your personality					

	Extremely	Very	Moderately	Slightly	Not at all
Lack of responsiveness from friends					
Nagging friends (e.g. a friend keeps seducing you)					
Poor health or abnormal time periods					
Nothing new exiting					
Poor/inappropriate recommendations					

4.12 Kindly list other factors that could cause a reduction/diminishing in your interest in social media

4.13 Which of the following topics would affect your social media presence if it were to be the main discussion point? *

- Politics (e.g., political figures, parties, governments)
- Religion and spiritualism (e.g. denominations, Jesus, scripture, Illuminati,)
- Public figures personality discussions
- Sports (e.g. football, teams, sports personality)
- Ethnicity
- Other:

4.14 How does the following factors at your work place affect your presence in social media?

	Extremely	Very much	Moderately	Slight;y	Not at all
Busy work schedule					

	Extremely	Very much	Moderately	Slight;y	Not at all
Social media is disallowed at work place	○	○	○	○	○
Travelling to work stations	○	○	○	○	○
Beating deadlines	○	○	○	○	○
Lack of internet at work	○	○	○	○	○

4.15 Kindly list other factors at the work place that affect your presence in social media

4.16 In terms of job schedule; during which periods are you mostly busy or engaged

- ☐ I have a fully packed job schedule everyday
- ☐ When there is a project
- ☐ When a deadline is approaching
- ☐ In the mornings
- ☐ In the evenings
- ☐ Other:

4.17 At job; could you be relocated to a remote area without internet?

- ○ Yes
- ○ No

4.18 If yes, how would you access social media sites while working from such areas?

- ○ I would access via cyber cafe once in town
- ○ I would use my mobile phone for access
- ○ I would wait until am relocated to other areas with internet
- ○ I would forget about social media

- Other:

4.19 How often are you relocated to such remote areas?

- Never
- Rarely
- Sometimes
- Often
- Always

Part 5: Social Media Activities

5.1 If you are using these Social sites, please indicate how often you access the following services and features? *

	Every time am online	Daily	Between 2 - 5 times a week	Weekly	Monthly
Chat with Friends					
Post Comments					
Update Status					
Upload images					
Upload Videos					
Download items					
View other peoples' profiles					
Repost Other people's status					

5.2 On average, how much data do you upload to Social sites such as Facebook in a day? *

- None
- Can't tell

- Less than 1Mb
- 1Mb - 5Mb
- 6Mb - 25Mb
- Over 25Mb

5.3 On average how much data do you download from Social sites such as Facebook in a day

- None
- Can't tell
- Less than 1Mb
- 1Mb - 5Mb
- 6Mb - 25Mb
- More than 25Mb

5.4 On average how often do you update your Social media Status (e.g. Facebook status, Post a Tweet)? *

- Less than once a day
- Daily
- 2 - 5 times a day
- 6 - 10 times a day
- 10 - 15 times a day
- more than 15 times a day

5.5 Do you experienced none responsiveness or unexpected exit by a friend when chatting? *

- Not At all
- Scarcely
- Sometimes
- Often
- Always

5.6 What is your immediate reaction when this happens *

- Exit social media all together
- Change to other activities away from social media
- Continue with other activities within social media
- Continue sending them chat messages

Part 6: Proposed Feature for Absence Recommendation

6.1 How would you rate the importance if the following services and features were made available in the Social Networks? *

	Not Important	Abit Important	Important	Very important	Absolutely Important
Warning if a friend is about to exit chat					
Suggestion on when a friend would be next available					
Suggestion on how long a friend is available					
Information on abnormal absence of a friend					
Recommendation of Friend based on period you knew them					

6.2 What would you expect to be recommended to you from your social website? *List any item you prefer

Thank you for your time and contribution in completing this questionnaire. Kindly send your completed questionnaire to this email address: icesiremulacs@gmail.com

0

Submit

Never submit passwords through Google Forms.