A
Major Project-II Report
On

A Context-Aware Recommender System Using the Sentiment Analysis of Tweets in the Environment of Internet of Things

Submitted in Partial Fulfillment of the Requirement for the Degree of

MASTER OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

MAYUR DAS
2K14/CSE/10

Under the Esteemed guidance of
Mr. VINOD KUMAR

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Shahabad Daulatpur, Main Bawana Road,
Delhi-110042.
JUNE, 2016

CERTIFICATE

This is to certify that Major Project-II Report entitled **“A Context-Aware Recommender System Using the Sentiment Analysis of Tweets in the Environment of Internet of Things”** submitted by **Mayur Das, Roll No. 2K14/CSE/10** for partial fulfillment of the requirement for the award of degree Master of Technology (Computer Science and Engineering) is a record of the candidate work carried out by him under my supervision.

(Project Guide)

Mr. Vinod kumar
Associate Professor
Department Of Computer Science & Engineering
Delhi Technological University

DECLARATION

I hereby declare that the major Project-II work entitled "**A Context-Aware Recommender System Using the Sentiment Analysis of Tweets in the Environment of Internet of Things**" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of degree of Master Of Technology (Computer Science and Engineering) is a bonafide report of Major Project-1I carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Mayur Das
2K14/CSE/10

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my project supervisor Mr. Vinod Kumar for providing the opportunity of carrying out this project and being the guiding force behind this work. I am deeply indebted to him for the support, advice and encouragement he provided without which the project could not have been a success.

Secondly, I am grateful to Dr. O.P.Verma, HOD, Computer Science & Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this work to be carried out. Last but not the least I would like to express sincere gratitude to my parents and friends for constantly encouraging me during the completion of work.

Mayur Das
University Roll no: 2K14/CSE/10
M.Tech (Computer Science & Engineering)
Department of Computer Engineering
Delhi Technological University
Delhi – 110042

ABSTRACT

Currently recommender systems are incorporating context and social information of the user, producing context aware recommender systems. In the future, they will use implicit, local and personal information of the user from the Internet of Things; where anyone and anything will be connected at anytime and anywhere. A Context- Aware Recommendation System has been introduced in this thesis. The fact that the future is for Internet of Things, and the multiple recommendation leads to my system design, in which multi-type rather than one type of recommendations will be recommended to the user. In this paper, a design of a context aware recommender system that recommends different types of items under the Internet of Things paradigm is proposed. A major part of this design is the context aware management system. In this system, we have used a neural network that will do the reasoning of the context to determine whether to push a recommendation or not and what type of items to recommend. The neural network inputs are derived virtually from the Internet of Things, and its outputs are scores for three types of recommendations, they are: songs, movies and none. These scores have been used to decide whether to push a recommendation or not, and what type of recommendations to. The results of 1000 random contexts were tested. For an average of 98.80% of them, our trained neural network generated correct recommendation types in the correct times and contexts.

List of Figures

List of Abbreviations

RS	Recommender System
CF	Collaborative Filtering
CBF	Content–based Filtering
ANN	Artificial neural network
FFNN	Feed- forward neural network
RBNN	Radial-basis neural network
MLP	Multi- layer perceptron neural network
CAMS	Context- Aware Management System

TABLE OF CONTENTS

CHAPTER 3: SENTIMENT ANALYSIS

CHAPTER 4: INTERNET OF THINGS

CHAPTER 5: ARTIFICIAL NEURAL NETWORK

CHAPTER 1

Introduction

1.1. BACKGROUND AND MOTIVATION

Many businesses nowadays embed recommendation systems in their web sites, in order to study the tastes of their customers, and achieve some business objectives. Among other objectives we have:

- the increase of traffic to their web site,
- the elaboration of marketing policies tailored to their customers' tastes,
- or simply the promotion of a given product.

Currently recommender systems are incorporating context and social information of the user, producing context aware recommender systems. In the future, they will use implicit, local and personal information of the user from the Internet of Things; where anyone and anything will be connected at anytime and anywhere. The fact that the future is for Internet of Things, and the emergence of proactivity concept leads to our system design, in which multi-type rather than one type of recommendations will be recommended proactively to the user in real time. In this paper, a design of a context aware recommender system that recommends different types of items under the Internet of Things paradigm is proposed. A major part of this design is the context aware management system. In this system, we have used a neural network that will do the reasoning of the context to determine whether to push a recommendation or not and what type of items to recommend. The neural network inputs are derived virtually from the Internet of Things, and its outputs are scores for two types of recommendations, they are: songs and movies. The results of 1000 random contexts were tested. For an average of 98.80% of them, our trained neural network generated correct recommendation types in the contexts.

1.2. OUTLINE OF THESIS

The outcome of this thesis will consist of five major parts.

The first part will be a summary of the researches that had been done on the recommendation (section 2), sentiment analysis (section 3) and Internet of Things (section 4) subjects. Section 2 includes the definitions and goals of recommender systems. Moreover it provides the survey on existing recommendation methods with the description of their advantages and disadvantages. Section 3 describes the concept and the main features of sentiment analysis as well as proposition of classification of these analyses. It provides the existing classification on the sentiment analysis.

Firstly, the data that will be used in recommendation process are identified and based on it user profile, which consists of several components, is proposed. After that the recommendation process is presented. It consists of many elements such as data preparation, calculation of the similarity between users choice. The precise description of each of these stages is provided.

The second part describes the challenges of the proposed method. The main subjects considered there are the rules and adjustment of weights, when the similarity between users choice is calculated. Next, third part contains section in which “how it works” examples are presented.

Next, the possible future improvements of the framework are shown. One of the elements that require further work is the recalculation of weights during the calculation of the final similarity function based on the users feedbacks.

The last stage of this thesis is the conclusions that appeared during the research on the framework. This part shows what the added value of proposed framework for the world of science is.

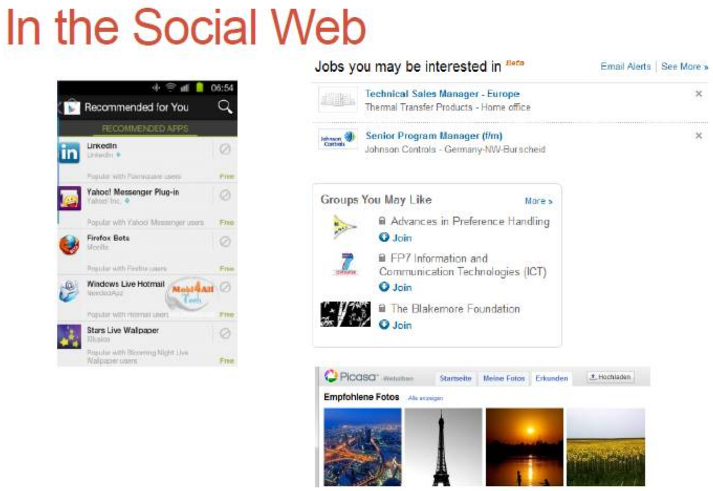
CHAPTER 2

Literature Review

2.1. RECOMMENDER SYSTEMS

There are many researchers who have worked in Context- Aware Recommender System. The information on Internet is growing rapidly and the visitors also required some system to recommend them the best information which they required from the vast grown information on internet. Here the recommender system comes into play. Recommendation systems changed the way inanimate websites communicate with their users. It is an intelligent system for the user to suggest them items that they may be interested in. Recommender systems are designed to help user/s in finding their items of interest from large collections of items such as movies, news, books and magazines. The end product of a recommendation algorithm is a top-list of items recommended for the user which in turn is ordered by an evaluated score representing the preference of that item for the user. The interest of a user in an item is assumed to be dependent on the value of the item being recommended, i.e., highest the value, more interested the user will be. Producing the right recommendations is not a trivial matter and there are several studies and research on evaluating the recommendations of such a system. Recommender systems are usually classified into two broad categories: Content based and Collaborative Filtering. In content-based recommendation [1], [2] one tries to recommend items similar to those a given user has liked in the past, whereas in collaborative recommendation [3], [2] one identifies users whose tastes are similar to those of the given user and recommends items they have liked. For instance, a content-based recommendation would be some- thing like Movie X is recommended because its category is Action and contains the term Bruce Willis, which are features contained in article you rated. A collaborative recommendation on the other hand would be like Movie X is recommended because other users similar to you have liked it. For example, if Bob and Wendy liked the same movies as you in the past and they both rated

Star Wars highly, you might like it, too. Recommender systems are a useful alternative to search algorithms since they help users discover items they might not have found by themselves. Interestingly enough, recommender systems are often implemented using search engines indexing non-traditional data.



In the Social Web

06:54
Recommended for You
RECOMMENDED APPS

LinkedIn
LinkedIn
Popular with Foursquare users
Free

Yahoo! Messenger Plug-in
Yahoo! Inc.
Popular with Yahoo! Messenger users
Free

Firefox Beta
Mozilla
Popular with Firefox users
Free

Windows Live Hotmail
NeededApz
Popular with Hotmail users
Free

Stars Live Wallpaper
Illusion
Popular with Blooming Night Live Wallpaper users
Free

Jobs you may be interested in Beta
Email Alerts | See More »

Technical Sales Manager - Europe
Thermal Transfer Products - Home office

Senior Program Manager (f/m)
Johnson Controls - Germany-NW-Burscheid

Groups You May Like
More »

Advances in Preference Handling
Join

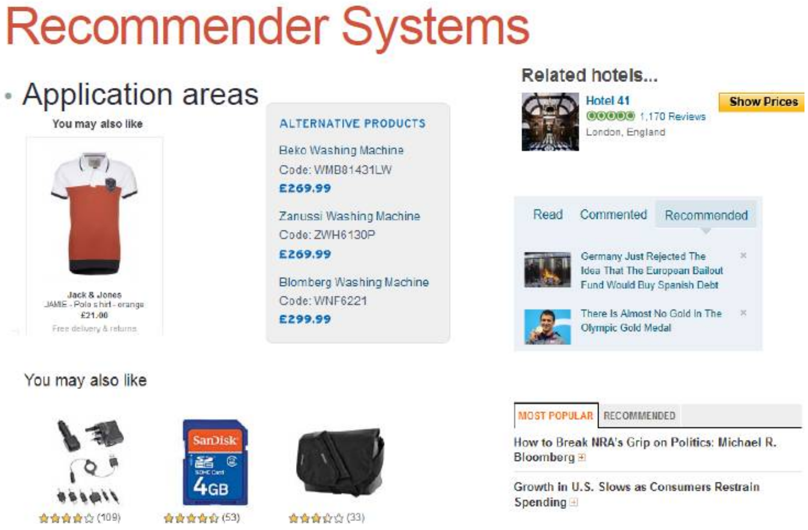
FP7 Information and Communication Technologies (ICT)
Join

The Blakemore Foundation
Join

Empfohlene Fotos
Alle anzeigen

Figure 1: Social Web Recommender System

These days recommender systems are widely being used in e-commerce websites [4] and are also gaining much popularity within the academic research community where many algorithms have been developed for providing recommendations.



Recommender Systems

- Application areas

You may also like

Jack & Jones
JAMIE - Polo shirt - orange
£21.00
Free delivery & returns

ALTERNATIVE PRODUCTS

Beko Washing Machine
Code: WMB81431LW
£269.99

Zanussi Washing Machine
Code: ZWH6130P
£269.99

Blomberg Washing Machine
Code: WNF6221
£299.99

Related hotels...

Hotel 41
1,170 Reviews
London, England
Show Prices

Read | Commented | Recommended

Germany Just Rejected The Idea That The European Bailout Fund Would Buy Spanish Debt

There Is Almost No Gold In The Olympic Gold Medal

You may also like

MOST POPULAR | RECOMMENDED

How to Break NRA's Grip on Politics: Michael R. Bloomberg

Growth in U.S. Slows as Consumers Restrain Spending

Figure 2: E-Commerce Recommender System

In such a scenario, an application designer who wishes to use a recommender system for his application must choose between candidate algorithms. Typically, such decisions are based on experimental results that compare the performance of these candidate algorithms. Generally, such performance evaluations are carried out by applying some evaluation measures [5]. From the literature it can be seen that there are several measures to evaluate recommender systems. For instance, Mean Absolute Error (MAE) is used to evaluate accuracy of predicted ratings. The problem with mean absolute error is that it is less appropriate when the granularity of true preference is small [6]. Precision and Recall are also used to evaluate the utility of recommendations produced by recommender systems. These measures do not attempt to directly measure the ability of an algorithm to accurately predict ratings and are sensitive to the number of recommendations k. Similarly, other measures such as coverage, novelty, serendipity and diversity are used to judge quality of recommendation from different perspectives but as is well known, most of these evaluation measures need online user studies. It is surprising that the measures mentioned above fail to address certain important issues related to recommendation. For instance, what happens when an already recommended item is added to the user profile? None of the evaluation measures for recommender systems takes into account the number of items from the previous recommendation that are continued to be recommended (retained) in the next recommendation when the same algorithm is run with an addition of an already recommended item.

2.2. DEFINITION OF RECOMMENDER SYSTEMS

There are many definitions of recommender systems. One of the first was presented by Paul Resnick and Hal R. Varian in 1997. They claim that “in a typical recommender system, people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients” [16].

These systems are usually defined in terms of their functionality as the systems or agents that suggest the products to the users who purchase products on e–commerce sites. The recommender systems help the consumer to make the decision what to buy.

The recommender systems can be classified because of the level of personalization into non–personalized and personalized methods [10] (Figure 3). The former methods do not consider the characteristics and preferences of the customers, whereas the latter tightly depends on the user profile. The individual methods that are enumerated in Figure 3 will be described in the next section.

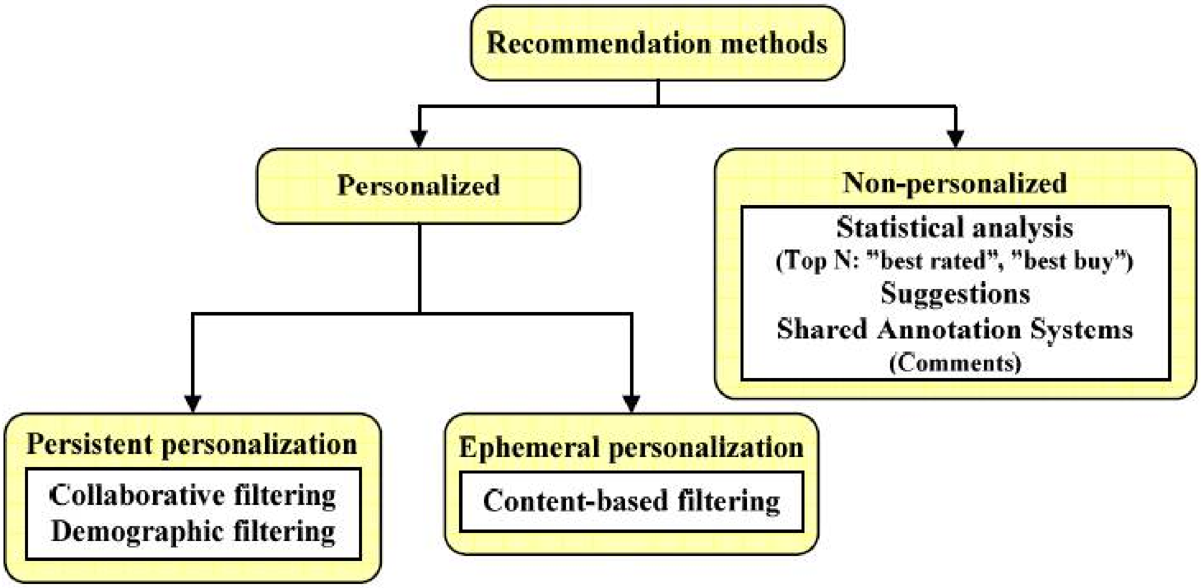


Figure3: Levels of the personalization in the recommender systems

The example of the non–personalized method is the recommendation that suggests the products which were best rated in the past by all the customers in average (“best rated”) or the number of their copies, which were sold, is the greatest (“best buy”). In order to create this kind of recommendation the statistical methods are used commonly. Moreover, in another variant of non–personalized approach even new items can be recommended, e.g. the books published in the last month. This kind of recommendation depends on the policy of the e– commerce and belongs to the techniques where not much calculation is required. The main feature of those suggestions is that they are the same for all customers. Usually, it is easy and quite convenient for a user to find one item from the list of the most popular ones. This is similar to the situation, when someone goes to the bookstore and finds there the shelf with the bestsellers.

However, some research claims that the recommender systems are only those ones, which produce personalized recommendations [8]. In other words, the output of these systems is the individualized recommendation that helps to guide the single user to products Recommendation system for online social network or services that fulfill their particular needs. As a result, they cope with information overload better than the non–personalized methods and enable to find and purchase the right items from the large amount of possible choices. The personalized recommendation is based either on the demographic information about users or on the analysis of the past behavior of the user in order to predict their future behavior (collaborative and content–based filtering) [9].

Moreover, the personalization can be either persistent or ephemeral [9]. Persistent personalization, based on the previous users' behaviors, enables to create unique list of products for each user. The requirement that ought to be fulfilled in this situation is that customers must log into the system in order to create user profile for each of them. In a persistent personalized recommendation each person on the Web site sees different recommendations because they depend directly on the users' personal data. The recommendations rely on the information derived from the survey responses, purchasing history, products ratings, etc. The user profile is not necessary in the ephemeral personalization. In this case the recommendations are created according to the users' behaviors during a current session, their navigation and selection [9]. In this technique the recommendations are the same for all users [10]. One of the formal definitions of the recommendation problem was formulated in [8]. Authors claim that the recommendation problem is to predict how the users will rate the products that they have not seen yet. When the system estimates the ratings it can recommend to the user the items with the highest rating. The formulation of the recommendation problem can be presented as [8]:

$$\forall c \in C, \ \ s'_c = \arg \max_{s \in S} u(c, s). \qquad \textbf{(1)}$$

where:

C – The set of all users

S – The set of all items *s* that can be recommended

S_c' – The product that has not been seen yet by user *c* and it has the highest estimated rating from all items unrated by user *c*

u(*c*, *s*) – The measurement called utility function that calculates the usefulness of item *s* to user *c*. This can be seen as the formal definition of the personalized recommendation (formula 1).

2.3. GOALS OF RECOMMENDER SYSTEMS

Recommender systems became an important and almost integral part of recent web sites; what is more, the vast number of them is applied to e–commerce. Jeff Bezos, CEO of Amazon.com, said: “If I had 3 million customers on the Web, I should have 3 million stores on the Web” [9]. Why do people believe that personalization and recommendations are a crucial part of e–commerce? The aim of these systems is to help the potential buyers to pick the appropriate product to buy, so that they can be seen as decision support systems. On the other hand, they serve as the marketing help for the e–commerce stores because they increase the attractiveness of the offer.

The main goals of the recommender systems are as follow:

- To cope with information overload
- To help all customers (new, frequent, and infrequent) to make decisions what
- products to buy, which news to read next , which movie is worth watching, etc.
- To convert observers to buyers
- To build credibility through community [9] and maintain the loyalty of the customers
- To inviting customers to come back [9]
- To enhance e–commerce sales and cross–sell [9]

The first two items show why the RS are important from the consumer point of view. First of all, they are very useful tool that help to cope with the information overload. The recommender systems enable to select a small subset of items, from millions of products, that seems to fit the users’ needs and preferences. Although it is almost impossible to predict precisely the users’ needs, such set of suggestions helps to limit the number of choices. Furthermore, by restricting the number of suggested products, these kinds of systems help people to make decisions, what items to buy, which news to read next [12] or which movie is worth watching, much faster than by the regular look through. The rest of the enumerated above items show that RS can be seen as the marketing tools because they enhance e–commerce sales [9]. As it was mentioned before, these systems can help people to find the products that they want to have. As a result this facilitates to convert the people who only watch to the buyers. When consumers buy things that are recommended by the system, the additional items can be suggested in order to increase the cross–sell. This leads to building and maintaining the loyalty of the customers [10], what is more, it encourages the customer to come back in the future. In the Internet and e–commerce where the number of competitors is very high, this feature is a crucial advantage of the recommender systems [9].

The aim of all the goals that were pinpointed above is to satisfy the customer. The reason is simple. The researches show that it is much less expensive to keep a current customer than to find a new one. Moreover, the dissatisfied customer tends to complain about product or service to twice as many people as satisfied customers will tell positive things about the service or product [11].

Additionally, RS ought to be as high efficiency as possible in order to increase their ROI (Return on Investment). However, the recommendations not only should exist but also ought to be relevant. The problem that can appear is too high number of false–positive recommendations, which are defined as suggestions that were created for the users, although they do not suit them. In conclusion, the goals of the recommendations can be achieved only if the generated suggestions are relevant.

2.4. CATEGORIES OF RECOMMENDATION SYSTEMS

2.4.1. TAXONOMIES OF RECOMMENDER SYSTEMS

There are lots of taxonomies of recommender systems. They can be divided according to the fact whether the created recommendation is personalized or not [10]. The example of personalized method is collaborative filtering whereas the example of non– personalized technique is the statistical analysis (see Figure 3).

Some research distinguishes three main categories of RS, where all of them are personalized, and they are as followed: collaborative filtering, content–based filtering, and hybrid methods [8] (Figure 4). Adomavicius and Tuzhilin claim that these three categories are the most popular and significant recommendation methods. However, they pinpoint the shortcomings of those methods and propose possible improvements.

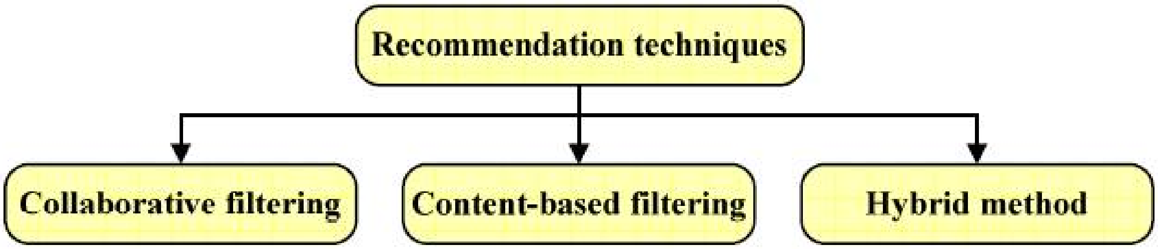


Figure 4 The example of taxonomy of the recommender systems [8]

Another research shows that RS can be divided into different groups because of the different criteria. Robin Burke distinguished five techniques of the recommendation (Figure 5) according to the type of a background and input data as well as the algorithm that is used to create the suggestions.

The background data is the information that the system possesses before the process of recommendation begins, whereas the input data enables to create the recommendations for particular user. The input data is provided by users and directly related to the user for whom recommendations are generated. The background data is the basis that enables to distinguish the following methods of recommendation: collaborative, content–based, demographic, utility–based, and finally knowledge–based techniques [13].

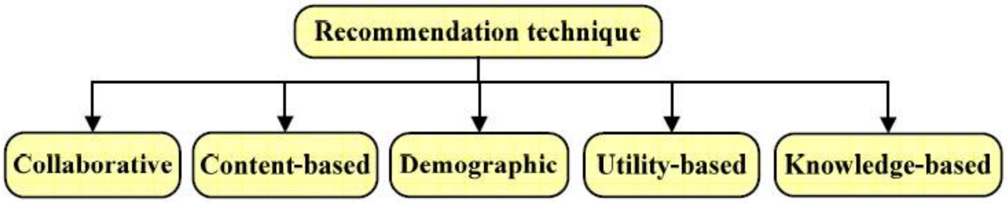


Figure 5: The example of taxonomy of the recommender systems [13]

In collaborative filtering, the background data is the set of all ratings of the items that the store possesses made by all customers of the store. The input data is the information about the ratings of the items in the store made by the single person for whom the system creates the recommendation. Finally, the algorithm identifies those users from that are similar to user for whom the suggestion is prepared and recommends these products, which were highly rated by the recognized similar users [13].

The rest of the enumerated above methods are characterized by Robin Burke in the analogical way and these characteristics are presented in Table 2.1.

Table 2.1 The division of recommendation techniques [13]

Technique	**Background data**	**Input data**	**Technique**
Collaborative	Rating from C[1] of items in S[2]	Ratings from C[3] of items in S	Identify users in C similar to C and extrapolate from their ratings of S[4]
Content-based	Features of items in S	Rating of user C of items in S	Generate a classifier that fits user's C rating behavior and use it on S
Demographic	Demographic information about C and their ratings of items in S	Demographic information about C	Identify users that are demographically similar to C and extrapolate from their ratings of S
Utility-based	Features of items in S	A utility function over items in S that describes preferences of C	Apply the function to the items and determine rank of S
Knowledge-based	Features of items in S. Knowledge of how these items meet user's needs	A description of user's C needs or interests	Infer match between S and user's C needs

Another research, that is worth to mention, is the taxonomy of recommender agents proposed by Miquel Montaner, Beatriz Lopez, and Lluis de la Rosa [14]. In their research authors distinguish two main approaches to the problem of RS: spatial and functional. Furthermore, from the functional point of view, they create eight dimensions, which are the basis for further classification of the recommender agents. Five of them concern the profile creation and maintenance, and three of them users` profile exploitation [14]. Although the profile creation and maintenance are very important parts of the recommender systems, they are out of the scope of this master thesis. The assumption is that the input data for the recommendation technique is the appropriate user profile. Figure 6 presents the dimensions that characterize RS.

The information filtering, user profile – item matching, and user profile matching are three main dimensions of the profile exploitation. Concerning information filtering techniques the most important techniques are demographic, content–based, and collaborative filtering. The goal of the user profile – item matching is to compare the representation of the user profile (e.g. user interests) and the description of the item and as the result pick the items that are relevant for the specific customer. The examples of such techniques are presented in the Figure 6. The last distinguished dimension is the user profile matching that enables to find the similar users or group of users.

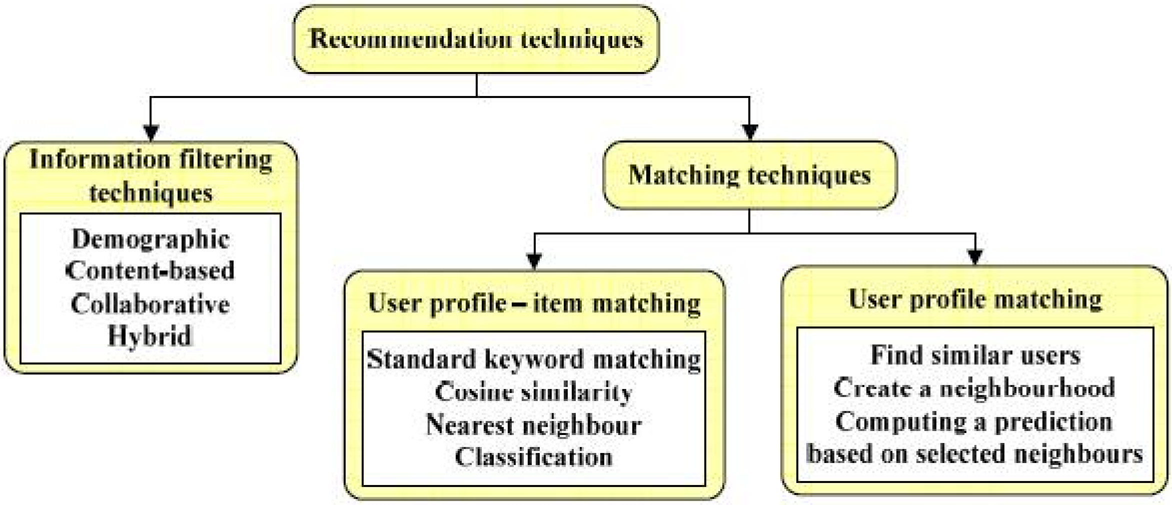


Figure 6: The example of taxonomy of the recommender systems [14]

Schafer et al. consider and analyze not only the recommendation methods, but also, similar to Burke, the input data that is delivered by the targeted customer (this for whom the recommendation is created) and by the rest of the customers [15]. This data serves as the input for the recommendation technique (Figure 7).

As the result of applying the appropriate technique the targeted customer receives the suggestion which item to buy. However, the output of applying recommendation method is not only the suggestions, but also the ratings and the prediction. The ratings are commonly used when the number of customers is small and the users know each other. In such case, it can be helpful to display the individual ratings of other customers [15]. Several RS provide the prediction of the ratings that the user would probably give to a product [15].

According to Schafer et al. the following types of the recommender systems can be distinguished: raw retrieval (called “null recommendation”), manually selected, statistical summarization, attribute–based, item–to–item correlation (also called content–based filtering), and user–to–user correlation (also called collaborative filtering) (Figure 7).

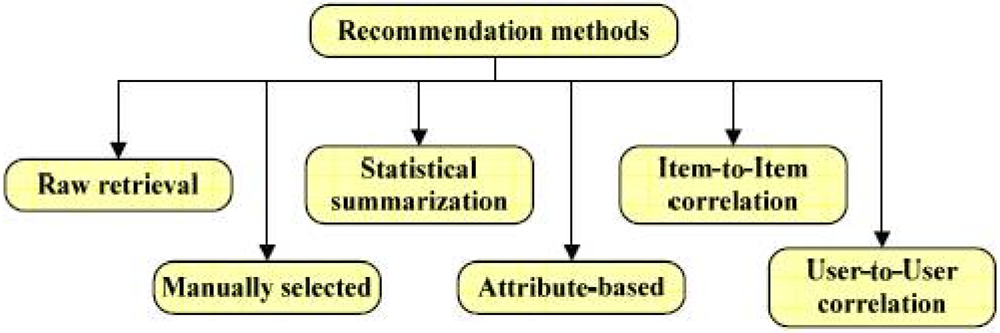


Figure 7: The example of taxonomy of the recommender systems [15]

2.4.2. SIMPLIFIED APPROACHES

To the simplified approaches belong statistical analysis and non–calculation techniques. Both of them do not require the complicated computations and are the non–personalized methods of recommendation.

The statistical analysis, in contrary to the non–calculation techniques requires calculations, which are, nevertheless, not very complicated. The system provides the ratings of the products that are based on the statistical factors. Some of these factors are: the number of sold units of each of the products and average rating of the product submitted by the customers who have already bought this product [17]. The statistics are calculated in the context of the whole community.

Advantages

- These methods do not require the complex calculations and this gives the opportunity to create the recommendations online

Disadvantages

- The recommendations are the same for all the users and in consequence the suggestions are too general and not personalized. As a result it is not possible to provide the recommendations fitting to the unique preferences of some users.

2.4.3. DEMOGRAPHIC FILTERING

In the demographic filtering (DF) method the users are divided into demographic classes in terms of their personal attributes. These classes serve as the input data to the recommendation process [13]. The goal of this process is to find the classes of people who like a certain product [18]. If people from class *C* like product *s* and there is person *c* (this user belongs to class C), who has not seen yet product *s*, then this product can be recommended to person *c*.

The customers provide the personal data via surveys that they fill in during the registration process [14, 17] or can be extracted from the purchasing history [17]. The first well known recommender system which utilized DF was Grundy [20] that suggested to users books. The sources of demographic data about the users were the interactive dialogues. “The user profiles are created by classifying users in stereotypical descriptions, representing the features of classes of users” [14]. Also some more recent research in the recommendation field applied this approach. For example LifeStyle Finder uses a demographic system PRIZM. This system divides the population of U.S.A according to the people survey responses, lifestyle characteristics, and purchasing history into 62 classes and for each class prepares the recommendations separately [19]. Pazzani proposed to use the machine learning that minimizes the effort required to gather the knowledge about users and create the classes of users. He classified users not only in terms of the data from the structural database, but also in terms of the text classification (web pages of the users) [18].

Advantages

- This technique may not require collecting the complex data such as history of users' purchases and ratings [13]

Disadvantages

- The classification can be too general and this leads to loose the individuality of the users. As the result the recommendations are too general [14, 17].
- In the Pazzani's experiment 57,7% of recommendations were correct [18]
- This method uses data that are provided by users. This data can be either incomplete or untrue [14]
- The classification is created according to the customers' interests, which tend to change over time. The demographic filtering does not support the adoption of the user profile to changes [14, 17]

2.4.4. COLLABORATIVE FILTERING

Technique that is the most mature and most widely used for RS is collaborative filtering (CF) [14]. It relies only on opinions explicitly delivered by the users on items [17]. The system recommends to the targeted customer products (or people), which have been evaluated in plus by another people, whose ratings are similar to the ratings of the targeted user [17]. The requirement that must be fulfilled is that the customer must log into the system in order to create for him the user profile. The representation of the user profile can be the vector of products and the ratings that were assigned to the particular items. The vector changes when the user rates the item [14]. The ratings can have the Boolean value (the customers like or dislike the item) or the value from the wider scale [14], e.g. from the range [–5, 5].

More formally, "the utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(cj, s)$ assigned to item s by those users $cj€C$ who are similar to user c" [8]. The utility function that was defined in section 2.2 in the formula 1 enables to calculate the usefulness of item s to user $c.$ The product for which the value of the utility function is the biggest is recommended to user c.

The process of collaborative recommendation consists of two main phases:

- Searching for the users that are similar to the current user (this one who is supposed to receive the recommendation). In the traditional CF the similarity between users is calculated according to their personal ratings [17, 14]. In the other words, the ratings of the targeted user (this for whom the recommendation is created) are compared to other users' ratings, in order to find the users similar to the targeted client. However, there are many other ways to compute this similarity.
- Recommending the items that are high graded by users who were found in the first phase. One of the methods is to recommend items that were high rating or most often buying (in this case it utilized the "Top N" method).

Although the classic version of this technique was based on ratings the products, now there are many variations of it. There are two main variants of CF. The first one is the k– nearest neighbors and the second one is the nearest neighborhood. The former one was used in the early recommender systems [24].

Collaboration filtering system searches for similar users (nearest neighbor) or group of users (nearest neighborhood) and then uses ratings from this set of users (groups of users) to predict items that will be liked by the current user.

There have been many RS which use the collaborative filtering method either in the academia or in the industry environment. The electronic mail was one of the first areas where the CF was used and the system was called Tapestry [21]. Other systems that first utilized this technique to automate prediction were GroupLens that calculated the correlation between the users of Usenet newsgroups [22], Ringo [23], and Video Recommender [25].

Advantages

- Although it is the oldest recommendation method, it is still very effective
- This method does not require providing the representation of the object that can be easily read by the computers [13]

Disadvantages

- Problems with new users (cold start), and new products (the early–rated problem [14]) that have not been yet rated by the users. Many people have to rate the products before the system will be effective [17]
- Data sparseness is the problem which can occur when there are many items to rate; the set of items changes very often and the number of customers is relatively small [17, 14]. It impedes finding users similar to the target one
- Difficulty in spotting “unpredictable” users with rare preferences and unusual opinions about products [17]
- In traditional collaborative filtering systems the amount of work increases simultaneously to the number of participants and items in the system. The computation method is quite complex, so it is usually done offline. Moreover, it is expensive because it requires gathering a lot of data to be effective

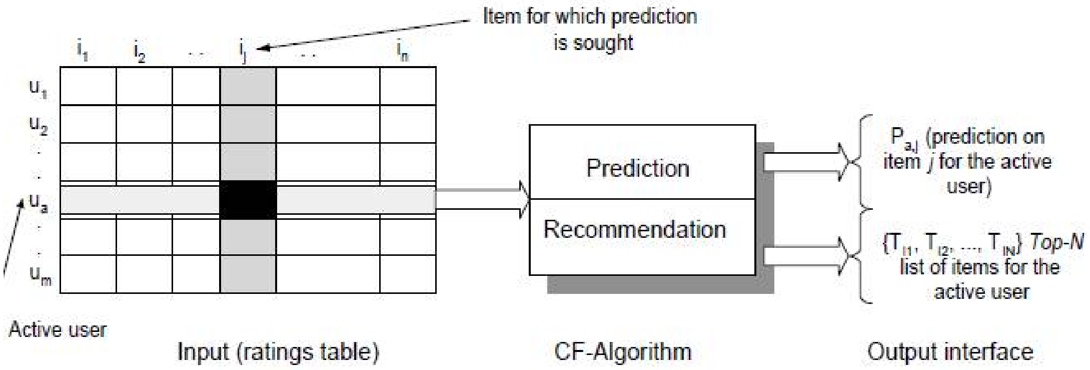

Item for which prediction is sought
i_1
i_2
i_j
i_n
u_1
u_2
u_a
u_m
Active user
Input (ratings table)
Prediction
Recommendation
CF-Algorithm
$P_{a,j}$ (prediction on item j for the active user)
$\{T_{i1}, T_{i2}, ..., T_{iN}\}$ Top-N list of items for the active user
Output interface

Figure 8: The Collaborative Filtering Process

2.4.4.1. OVERVIEW OF THE COLLABORATIVE FILTERING PROCESS

The goal of a collaborative filtering algorithm is to suggest new items or to predict the utility of a certain item for a particular user based on the user’s previous likings and the opinions of other like-minded users. In a typical CF scenario, there is a list of *m* users U = {*u*1*, u*2*, . . . , um*} and a list of *n* items I = {*i*1*, i*2*, . . . , in*}. Each user *ui* has a list of items *Iui* , which the user has expressed his/her opinions about. Opinions can be explicitly given by the user as a *rating score*, generally within a certain numerical scale, or can be implicitly derived from purchase records, by analyzing timing logs, by mining web hyperlinks and so on [28]. Note that $I\ ui \in I$ and it is possible for Iui to be a *null-set*. There exists a distinguished user $ua \in U$ called the *active user* for whom the task of a collaborative filtering algorithm is to find an item likeliness that can be of two forms.

- **Prediction** is a numerical value, $P_{a,\ j}$, expressing the predicted likeliness of item $i\ j$ _$\in I_{ua}$ for the active user u_a. This predicted value is within the same scale (e.g., from 1 to 5) as the opinion values provided by $u\ _a$.
- **Recommendation** is a list of N items, $I_r \subset I$, that the active user will like the most. Note that the recommended list must be on items not already purchased by the active user, i.e., $I_r \cap I_{ua} = \varphi$. This interface of CF algorithms is also known as *Top-N recommendation*.

Figure 8 shows the schematic diagram of the collaborative filtering process. CF algorithms represent the entire $m \times n$ user-item data as a ratings matrix, A. Each entry $ai,\ j$ in A represent the preference score (ratings) of the i th user on the j th item. Each individual ratings is within a numerical scale and it can as well be 0 indicating that the user has not yet rated that item. Researchers have devised a number of collaborative filtering algorithms that can be divided into two main categories—*Memory-based (user-based)* and *Model-based (item-based)* algorithms [6]. In this section we provide a detailed analysis of CF-based recommender system algorithms.

Memory-based Collaborative Filtering Algorithms Memory-based algorithms utilize the entire user-item data-base to generate a prediction. These systems employ statistical techniques to find a set of users, known as *neighbors*, that have a history of agreeing with the target user (i.e., they either rate different items similarly or they tend to buy similar set of items). Once a neighborhood of users is formed, these systems use different algorithms to combine the preferences of neighbors to produce a prediction or *top-N* recommendation for the active user. The techniques, also known as *nearest-neighbor* or user-based collaborative filtering are more popular and widely used in practice.

Model-based Collaborative Filtering Algorithms Model-based collaborative filtering algorithms provide item recommendation by first developing a model of user ratings. Algorithms in this category take a probabilistic approach and envision the collaborative filtering process as computing the expected value of a user prediction, given his/her ratings on other items. The model building process is performed by different *machine learning* algorithms such as **Bayesian network, clustering,** and **rule-based** approaches. The Bayesian network model [6] formulates a probabilistic model for collaborative filtering problem. Clustering model treats collaborative filtering as a classification problem [2, 6, 29] and works by clustering similar users in same class and estimating the probability that a particular user is in a particular class *C*, and from there computes the conditional probability of ratings. The rule-based approach applies association rule discovery algorithms to find association between co-purchased items and then generates item recommendation based on the strength of the association between items.

2.4.4.2. CHALLENGES OF USER-BASED COLLABORATIVE FILTERING ALGORITHMS

User-based collaborative filtering systems have been very successful in past, but their widespread use has revealed some potential challenges such as:

- **Sparsity.** In practice, many commercial recommender systems are used to evaluate large item sets (e.g., Amazon. com recommends books and CDnow.com recommends music albums). In these systems, even active users may have purchased well under 1% of the items (1% of 2 million books is 20, 000 books). Accordingly, a recommender system based on nearest neighbor algorithms may be unable to make any item recommendations for a particular user. As a result the accuracy of recommendations may be poor.
- **Scalability.** Nearest neighbor algorithms require computation that grows with both the number of users and the number of items. With millions of users and

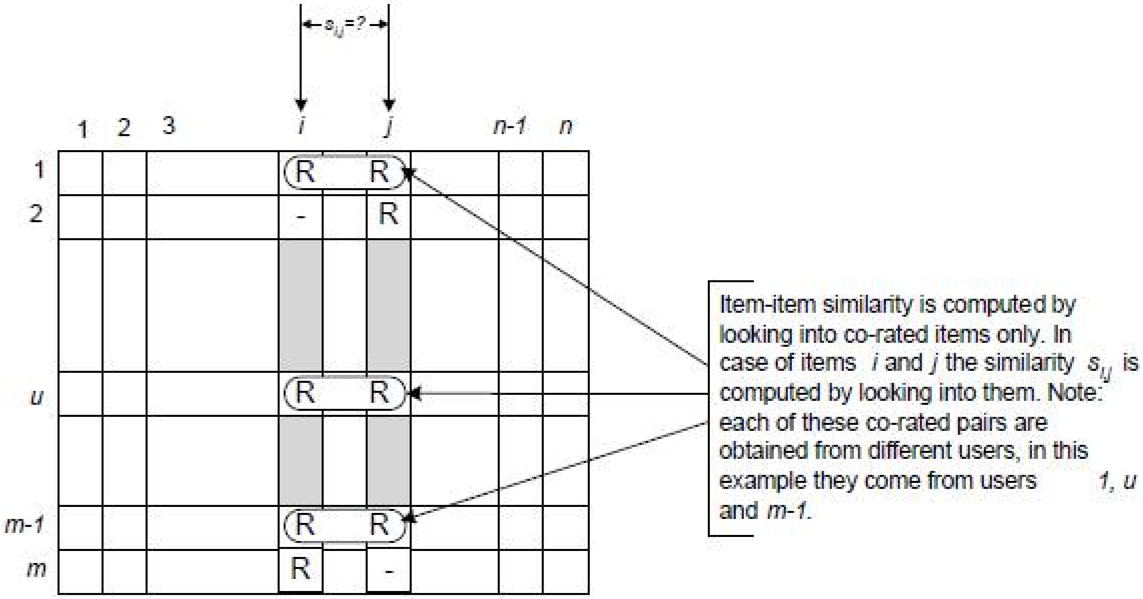
items, a typical web-based recommender system running existing algorithms will suffer serious scalability problems.

2.4.4.3. ITEM-BASED COLLABORATIVE FILTERING ALGORITHM

In this section we study a class of item-based recommendation algorithms for producing predictions to users. Unlike the user-based collaborative filtering algorithm discussed in Section 2 the item-based approach looks into the set of items the target user has rated and computes how similar they are to the target item *i* and then selects *k* most similar items {*i*1*, i*2*, . . . , ik* }. At the same time their corresponding similarities {*si*1*, si*2*, . . . , sik* } are also computed. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. We describe these two aspects namely, the similarity computation and the prediction generation in details here.

2.4.4.3.1. ITEM SIMILARITY COMPUTATION

One critical step in the item-based collaborative filtering algorithm is to compute the similarity between items and then to select the most similar items. The basic idea in similarity computation between two items *i* and *j* is to first isolate



Item-item similarity is computed by looking into co-rated items only. In case of items *i* and *j* the similarity $s_{i,j}$ is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users *1, u* and *m-1*.

Figure 9: Isolation of the co-rated items and similarity computation

the users who have rated both of these items and then to apply a similarity computation technique to determine the similarity $s_{i,\ j}$. Figure 9 illustrates this process, here the matrix rows represent users and the columns represent items. There are a number of different ways to compute the similarity between items. Here we present three such methods. These are cosine-based similarity, correlation-based similarity and adjusted-cosine similarity.

2.4.4.3.1.1. COSINE-BASED SIMILARITY

In this case, two items are thought of as two vectors in the *m* dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the *m* × *n* ratings matrix in Figure 9, similarity between items *i* and *j* , denoted by *sim(i, j)* is given by

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

where “·” denotes the dot-product of the two vectors.

2.4.4.3.1.2. CORRELATION-BASED SIMILARITY

In this case, similarity between two items *i* and *j* is measured by computing the *Pearson-r* correlation $corr_{i,\ j}$. To make the correlation computation accurate we must first isolate the co-rated cases (i.e., cases where the users rated both *i* and *j*) as shown in Figure 9. Let the set of users who both rated *i* and *j* are denoted by *U* then the correlation similarity is given by

$$sim(i,j) = \frac{\sum_{u\in U}(R_{u,i}-\bar{R}_i)(R_{u,j}-\bar{R}_j)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_i)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_j)^2}}.$$

Here $R_{u,i}$ denotes the rating of user u on item i , .$\bar{R}_i$ is the average rating of the i^{th} item.

2.4.4.3.1.3. ADJUSTED COSINE SIMILARITY

One fundamental difference between the similarity computation in user-based CF and item-based CF is that in case of user-based CF the similarity is computed along the rows of the matrix but in case of the item-based CF the similarity is computed along the columns i.e., each pair in the co-rated set corresponds to a different user (Figure 9). Computing similarity using basic cosine measure in item-based case has one important drawback–the difference in rating scale between different users are not taken into account. The adjusted cosine similarity offsets this drawback by subtracting

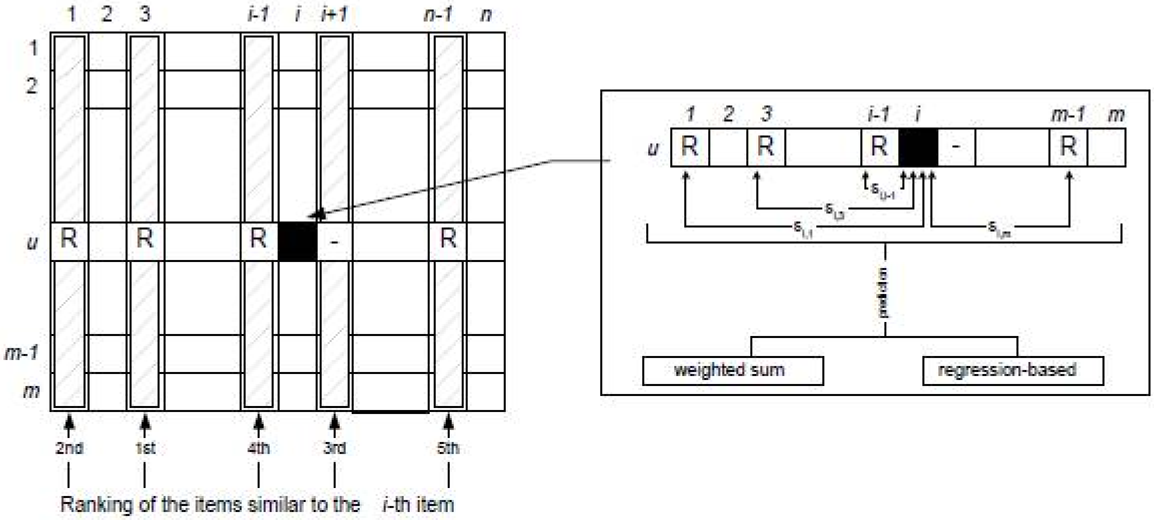


Figure 10: Item-based collaborative filtering algorithm. The prediction generation process is illustrated for 5 neighbors

the corresponding user average from each co-rated pair. Formally, the similarity between items *i* and *j* using this scheme is given by

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \bar{R_u})(R_{u,j} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R_u})^2}}.$$

Here $\bar{R}._u$ is the average of the *u*-th user's ratings.

2.4.4.3.2. PREDICTION COMPUTATION

The most important step in a collaborative filtering system is to generate the output interface in terms of prediction. Once we isolate the set of most similar items based on the similarity measures, the next step is to look into the target users ratings and use a technique to obtain predictions. Here we consider two such techniques.

2.4.4.3.2.1. Weighted Sum

As the name implies, this method computes the prediction on an item *i* for a user *u* by computing the sum of the ratings given by the user on the items similar to *i* . Each ratings is weighted by the corresponding similarity $s_{i,\, j}$ between items *i* and *j* . Formally, using the notion shown in Figure 10 we can denote the prediction $P_{u,i}$ as

$$P_{u,i} = \frac{\sum_{\text{all similar items, N}} (s_{i,N} * R_{u,N})}{\sum_{\text{all similar items, N}} (|s_{i,N}|)}$$

Basically, this approach tries to capture how the active user rates the similar items. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range.

2.4.5. CONTENT-BASED FILTERING

The content–based filtering (CBF) uses the description of the items that were previously watched or purchased by the customer and evaluated by them in a positive way. The system recommends the consumers the items similar to the items that they liked in the past [17, 14]. The user profile is created being based on the features that occur in the items positively rated by the user [13, 14] and contains users' tastes, needs, and preferences [8]. Content–based recommendation is usually applied to suggest documents, web pages, Usenet news messages and other text–based items [8, 17].

In this method, the described above utility function is defined as "the utility *u*(*c*,*s*) of item *s* for user *c* is estimated based on the utilities *u*(c_i, s_i) assigned by user *c* to items s_i € *S* that are similar to item *s*" [8]. Analogous to the CF the product for which the value of the utility function is the biggest is recommended to user *c*.

The CBF is utilized for example in the text recommender systems such as NewsWeeder [26]. Other system, in which the users rate the Web documents and assign them values from the binary “hot” and “cold” scale is Syskill & Webert [27]. These ratings serve as the basis for the calculation of the probabilities of the words being in hot or cold documents. The Web Watcher system, which aim is to recommend links on the Web pages that the user will maybe visit in the future, monitors the users’ behaviors and choices of links on the Web pages [28].

Advantages

- Only the analysis of the items that one independent user has seen or bought must be done. In contrary to CF, this technique is not so complex

Disadvantages

- Overspecialization – when the system recommends items that are similar to those which were highly rated by the client, the effect of overspecialization can occur [8, 14]. It means that the items suggested to the user will be very similar and the customer can be bored by the continuous watching of the documents with overlapping content

- Limited content analysis – in CBF each item is described by the features. It is not always possible to create the sufficient set of features. The retrieval of the information from the text document is relatively easy in comparison to other types of documents (graphical, audio or video documents). Moreover, if two documents contain the same terms and as a result have the same set of features, it is not possible to distinguish them. In such case the system is not able to differentiate the well–written document from a badly written one [8]
- New user problem – the method becomes effective when the user rates sufficient number of items. The reliable recommendation can be created only when the system has the exact knowledge about the users’ preferences and needs. [8, 17]

2.4.6. HYBRID METHOD

The hybrid approach to recommendation combines the content–based and collaborative filtering. The main goal of hybrid methods is to avoid the shortcomings of the two previously enumerated methods [8, 13]. There are many different ways to combine the content–based and collaborative filtering. The best known are [8]:

- Implement both methods separately and combine the outputs of these methods
- Add some of the content–based characteristics to the collaborative filtering
- Add some of the collaborative characteristics to the content–based filtering·
- Develop one model that applies both content–based and collaborative characteristics

These two approaches complement each other and contribute to the other's effectiveness [13].

CHAPTER 3

SENTIMENT ANALYSIS

3.1. GENERAL SENTIMENT ANALYSIS

Sentiment analysis has been done on a range of topics. For example, there are sentiment analysis studies for movie reviews [32], product reviews [33,34], and news and blogs [37]. Below some general sentiment analysis concepts are discussed.

3.2. DIFFICULTY OF SENTIMENT ANALYSIS

Research shows that sentiment analysis is more difficult than traditional topic based text classification, despite the fact that the number of classes in sentiment analysis is less than the number of classes in topic-based classification [35]. In sentiment analysis, the classes to which a piece of text is assigned are usually negative or positive. They can also be other binary classes or multivalued classes like classification into 'positive', 'negative' and 'neutral', but still they are less than the number of classes in topic-based classification. The main reason that sentiment analysis is more difficult than topic-based text classification is that topic-based classification can be done with the use of keywords while this does not work well in sentiment analysis [36].

Other reasons for difficulty are: sentiment can be expressed in subtle ways without any ostensible use of negative words; it is difficult to determine whether a given text is objective or subjective (there is always a _ne-line between objective and subjective texts); it is difficult to determine the opinion holder (example, is it the opinion of the author or the opinion of the commenter); there are other factors such as dependency on domain and on order of words [35]. Other factors that make sentiment analysis difficult are that it can be expressed with sarcasm, irony, and/or negation.

3.3. CLASSIFICATION AND APPROACHES

As elaborated in the introduction CHAPTER, sentiment analysis is formulated as a text-classification problem. However, the classification can be approached from different perspectives suited to the work at hand. Depending on the task at hand and perspective of the person doing the sentiment analysis, the approach can be discourse-driven, relationship-driven, language-model-driven, or keyword-driven. Some of the perspectives that can be used in sentiment classification are discussed briefly in the subsequent subsections.

3.3.1. KNOWLEDGE-BASED APPROACH

In this approach, sentiment is seen as the function of some keywords. The main task is the construction of sentiment discriminatory-word lexicons that indicate a particular class such as positive class or negative class. The polarity of the words in the lexicon are determined prior to the sentiment analysis work. There are variations to how the lexicon is created. For example, lexicons can be created by starting with some seed words and then using some linguistic heuristics to add more words to them, or starting with some seed words and adding to these seed words other words based on frequency in a text [36].For some domains of tasks, there are publicly available discriminatory word lexicons for use in sentiment analysis. http://twitrratr.com/ and http://www.cs.pitt.edu/mpqa/ are two examples. http://twitrratr.com/ provides sentiment lexicons for Twitter sentiment analysis.

3.3.2. RELATINSHIP-BASED APPROACH

Here the classification task can be approached from the different relationships that may exists in or between features and components. Such relationships include relationships between discourse participants, relationships between product features. For example, if one wants to know the sentiment of customers about a product brand, one may compute it as a function of the sentiments on different features or components of it.

3.3.3. LANGUAGE MODELS

In this approach the classification is done by building n-gram language models. Presence or frequency of n-grams might be used. In traditional information retrieval and topic-oriented classification, frequency of n-grams is shown to give better results. Usually, the frequency is converted to TF-IDF to take term's importance for a document into account. However, [32], in sentiment classification of movie reviews found that term-presence gives better results than term frequency. They indicate that uni-gram presence is more suited for sentiment analysis. But a bit later than [32], [33] found that bi-grams and tri-grams worked better than uni-grams in sentiment classification of product reviews.

3.3.4. DISCOURSE STRUCTURES AND SEMANTICS

In this approach, discourse relation between text components is used to guide the classification. For example in reviews, the overall sentiment is usually expressed at the end of the text [32]. As a result the approach to sentiment analysis, in this case, might be discourse-driven in which the sentiment of the whole review is obtained as a function of the sentiment of the different discourse components in the review and the discourse relations that exist between them. In such an approach, the sentiment of a paragraph that is at the end of the review might be given more weight in the determination of the sentiment of the whole review. Semantics can be used in role identification of agents where there is a need to do so. for example Manchester beat Liverpool is different from Liverpool beat Manchester.

3.4. SENTIMENT ANALYSIS TECHNIQUES

Using one or a combination of the different approaches in subsection 3.4, one can employ one or a combination of machine learning techniques. Specifically, one can use unsupervised techniques, supervised techniques or a combination of them.

3.4.1. UNSUPERVISED TECHNIQUES

In unsupervised technique, classification is done by a function which compares the features of a given text against discriminatory-word lexicons whose polarity are determined prior to their use. For example, starting with positive and negative word lexicons, one can look for them in the text whose sentiment is being sought and register their count. Then if the document has more positive lexicons, it is positive, otherwise it is negative. A slightly different approach is done by [36] who used a simple unsupervised technique to classify reviews as recommended (thumbs up) or not recommended (thumbs down) based on semantic information of phrases containing an adjective or adverb. He computes the semantic orientation of a phrase by mutual information of the phrase with the word 'excellent' minus the mutual information of the same phrase with the word 'poor'. Out of the individual semantic orientation of phrases, an average semantic orientation of a review is computed. A review is recommended if the average semantic orientation is positive, not recommended otherwise.

3.4.2. SUPERVISED TECHNIQUES

The main task here is to build a classifier. The classifier needs training examples which can be labeled manually or obtained from a user-generated user labeled online source. Most used supervised algorithms are Support Vector Machines (SVM), Naive Bayes classifier and Multinomial Naive Bayes. It has been shown that Supervised Techniques outperform unsupervised techniques in performance [32]. Supervised techniques can use one or a combination of approaches we saw above. For example, a supervised technique can use relationship-based approach, or language model approach or a combination of them. For supervised techniques, the text to be analyzed must be represented as a feature vector. The features used in the feature vector are one or a combination of the features in 3.4.4 subsection.

3.4.3. COMBINED TECHNIQUES

There are some approaches which use a combination of other approaches. One combined approach is done by [38]. They start with two word lexicons and unlabeled data. With the two discriminatory-word lexicons (negative and positive), they create pseudo-documents containing all the words of the chosen lexicon. After that, they compute the cosine similarity between these pseudo-documents and the unlabeled documents. Based on the cosine similarity, a document is assigned either positive or negative sentiment. Then they use these to train a Naive Bayes classifier. performance with their approach than approaches using lexical knowledge or training data in isolation, or other approaches that use combined techniques. There are also other types of combined approaches that are complimentary in that different classifiers are used in such a way one classifier contributes to another [39].

3.4.4. FEATURE ENGINEERING

Since most of sentiment analysis approaches use or depend on machine learning techniques, the salient features of text or documents are represented as feature vector. The following are the features used in sentiment analysis.

Term presence or term frequency: In standard Information retrieval and text classification, term frequency is preferred over term presence. However, Pang et al. (2002) [32], in sentiment analysis for movie reviews, show that this is not the case in sentiment analysis. Pang et al. claim that this is one indicator that sentiment analysis is different from standard text classification where term frequency is taken to be a good indicator of a topic. Ironically, another study by Yang et al. (2006) [40] shows that words that appear only once in a given corpus are good indicators of high-precision subjectivity.

Term can be either uni-grams, bi-grams or other higher-order n-grams. Whether uni-grams or higher-order n-grams give better results is not clear. [32] claim that uni-grams outperform bi-grams in movie review sentiment analysis, but Dave et al. (2003) [33] report that _bi-grams and tri-grams give better product-review polarity classification.

POS (Part of speech) Tags: POS is used to disambiguate sense which in turn is used to guide feature selection [41]. For example, with POS tags, we can identify adjectives and adverbs which are usually used as sentiment indicators [36]. But, Turney himself found that adjectives performed worse than the same number of uni-grams selected on the basis of frequency.

Syntax and Negation: Collocations and other syntactic features can be employed to enhance performance. In some short text (sentence-level) classiffcation tasks, algorithms using syntactic features and algorithms using n-gram features were found to give same performance [41]. Negation is also an important feature to take into account since it has the potential of reversing a sentiment [41].There are attempts to model negation for better performance [42]. Na et al. (2004) [34] report 3% accuracy improvement for electronics product reviews by handling negation. Note also that negation can be expressed in more subtle ways such as sarcasm, irony and other polarity reversers.

3.5. TWITTER-SPECIFIC SENTIMENT ANALYSIS

There are some Twitter-specific sentiment analysis studies. Twitter sentiment analysis is a bit different from the general sentiment analysis studies because Twitter posts are short. The maximum number of characters that are allowed in Twitter is 140. Moreover Twitter messages are full of slang and misspellings [43]. Almost all Twitter sentiment classification is done using machine learning techniques. Two good reasons for the use of machine learning techniques are 1) the availability of huge amount of Twitter data for training, and 2) that there is test data which is user-labeled for sentiment with emoticons (avoiding the cumbersome task of manually annotating data for training).

A Twitter sentiment analysis study by Go et al. (2009) [43] does a two-classed (negative and positive) classification of tweets about a term. Emoticons (for positive ':)', for negative ':(') were used to collect training data from Twitter API. The training data was preprocessed before it was used to train the classifier. Preprocessing included replacing user names and actual URLs by equivalence classes of 'URL' and 'USERNAME' respectively, removing repeated letters to 2 (huuuuuuungry to huungry), and removing the query term. To select useful uni-grams, they used such feature selection algorithms as frequency, mutual information, and chi-square method. They experiment with three supervised techniques: multinomial Naive Bayes, maximum entropy and support vector machines (SVM). The best result, accuracy of 84%, was obtained with multinomial Naive Bayes using uni-gram features selected on the basis of their MI score. They also experimented with bi-grams, but accuracy was low. They claim the reason for this low accuracy is data spareness. Incorporating POS, and negation into the feature vector of uni-grams does not also improve results.

The above experiment does not recognize and handle neutral tweets. To take into account neutral tweets, they collected tweets about a term that do not have emoticons. For test data, they manually annotated 33 tweets as neutral. They merged these two datasets with the training data and test data used in the above two-classed classification. They trained a three-classed classifier and tested it, but the accuracy was very low, 40%.

Another study by Barbosa and Feng (2010) [44] used a two-phased approach to Twitter sentiment analysis. The two phases are: 1) classifying the dataset into objective and subjective classes (subjectivity detection) and 2) classifying subjective sentences into positive and negative classes (polarity detection). Suspecting that the use of n-grams for Twitter sentiment analysis might not be a good strategy since Twitter messages are short, they use two other features of tweets: meta information about tweets and syntax of tweets. For meta-info, they use POS tags (some tags are likely to show sentiment, eg. adjectives and interjections) and mapping words to prior subjectivity (strong and weak), and prior polarity (negative, positive and neutral). The prior polarity is reversed when a negative expression precedes the word. For tweet syntax features, they use #(hashtag, @(reply), RT(retweet), link, punctuations, emoticons, capitalized words, etc. They create a feature set from both the features and experiment with machine learning technique available in WEKA. SVM performs best. For test data, 1000 tweets were manually annotated as positive, negative, and neutral. The highest accuracy obtained was 81.9% on subjectivity detection followed by 81.3% on polarity detection.

A very related study to this thesis was done by Pak and Paroubek (2010) [45]. They did a three-classed (positive, negative, neutral) sentiment analysis on Twitter posts. They collected negative and positive classes using emoticons (for positive: _:-)_ , _:)_, _=)_, _:D_ ,etc and for negative: _:- (_, _:(_, _=(_, _;(_ , etc.). For the neutral class, they took posts from Twitter accounts of popular news outlets (the assumption is news headlines are neutral).

After the data collection, they did some linguistic analysis on the dataset. They POS tagged it and looked for any differences between subjective (positive and negative) and objective sentences. They note that there are differences between the POS tags of subjective and objective Twitter posts. They also note that there are difference in the POS tags of positive and negative posts. Then they cleaned the data by removing URL links, user names (those that are marked by @), RT (for retweet), the emoticons, and stop words. Finally they tokenized the dataset and constructed n-grams. Then they experimented with several classifiers including SVM, but Naive Bayes was found to give the best result. They trained two Naive Bayes Classifiers. One of them uses n-gram presence, and the other, POS tag presence. The probability of a sentiment (positive, negative, neutral) of a Twitter post is obtained as the sum of the summation of the probabilities of n-gram presence and the summation of the probabilities of n-gram POS tags. Namely,

$$L(s|M) \quad = \quad \sum_{g \in N} log(P(g|s)) + \sum_{t \in T} log(P(t|s))$$

where G is a set of n-grams of the tweet, T is the set of POS tags of the n-grams, M is the tweet and s is the sentiment (one of positive, negative, and neutral). The sentiment with highest likelihood (L(s|M)) becomes the sentiment of the new tweet.

Pak and Paroubek (2010) [45] achieved best result (highest accuracy) with bigram presence. Their explanation for this is that bi-grams provide a good balance between coverage (uni-grams) and capturing sentiment expression patterns (tri-grams) [45]. Negation('not' and 'no') is handled by attaching it to the words that precede and follow it during tokenization. The handling of negation is found to improve accuracy. Moreover, they report that removing n-grams that are evenly distributed in the sentiment classes improves accuracy. Evaluation was done on the same test data used by Go et al. (2009) [43]. However, they do not explicitly put their accuracy in number other than showing it in a graph (in which it seems to approach 1) and stating it in words saying a very high accuracy.

CHAPTER 4

INTERNET OF THINGS

4.1. Internet of Things(IoT): Nowadays, around two billions people around the world use the Internet for browsing the Web, sending and receiving emails, accessing multimedia content and services, playing games, using social networking applications and many other tasks. While more and more people will gain access to such a global information and communication infrastructure, another big leap forward is coming, related to the use of the Internet as a global platform for letting machines and smart objects communicate, dialogue, compute and coordinate. It is predictable that, within the next decade, the Internet will exist as a seamless fabric of classic networks and networked objects. Content and services will be all around us, always available, paving the way to new applications, enabling new ways of working; new ways of interacting; new ways of entertainment; new ways of living. In such a perspective, the conventional concept of the Internet as an infrastructure network reaching out to end-users' terminals will fade, leaving space to a notion of interconnected ''smart' objects forming pervasive computing environments [6]. The Internet infrastructure will not disappear. On the contrary, it will retain its vital role as global backbone for worldwide information sharing and diffusion, interconnecting physical objects with computing/ communication capabilities across a wide range of services and technologies.

The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals.

Unquestionably, the main strength of the IoT idea is the high impact it will have on several aspects of everyday-life and behavior of potential users. From the point of view of a private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. In this context, domotics, assisted living, e-health, enhanced learning are only a few examples of possible application scenarios in which the new paradigm will play a leading role in the near future. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as, automation and industrial manufacturing, logistics, business/process management, intelligent transportation of people and goods.

By starting from the considerations above, it should not be surprising that IoT is included by the US National Intelligence Council in the list of six ‘‘Disruptive Civil Technologies” with potential impacts on US national power [7]. NIC foresees that ‘‘by 2025 Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more”. It highlights future opportunities that will arise, starting from the idea that ‘‘popular demand combined with technology advances could drive widespread diffusion of an Internet of Things (IoT) that could, like the present Internet, contribute invaluably to economic development”. The possible threats deriving from a widespread adoption of such a technology are also stressed. Indeed, it is emphasized that ‘‘to the extent that everyday objects become information security risks, the IoT could distribute those risks far more widely than the Internet has to date”.

The architecture supporting interconnected devices evolve further and find implementations in areas like logistics, farming, industry, home automation and many others are already a fact but the restrictions in terms of interconnection solutions from different vendors, communities and standard groups become more obvious. Referring to the business aspects, the IoT enables a plethora of new opportunities, disruptive business models and use case scenarios. In many cases those connected devices and objects are not Hypertext Transfer Protocol

(HTTP) driven and that is why there is a lack of decent application integration layers and the applications development is hard to be achieved.

Internet of Things is used to get the data form internet of the user for the recommendation. Twitter is one of the social networking sight where people use to Tweet about their life every minute. By analyzing the tweets of a user, the Mood of an specific user can be determined. The mood of user is used to suggest a specific genre of songs to user.

CHAPTER 5

ARTIFICIAL NEURAL NETWORK

5.1. What is a classifier?

When working with statistics and other areas where large amounts of data are collected and analyzed, it is often necessary to sort the data points into sub-groups. This can be a very hard task for a human, who often aren't able to recognize which class a data point belongs to because of the large amounts of data contained in each data point. Instead, a digital classifier is used.

There are several different methods of creating a digital classifier, and two of the most common are described below. Common for all classifiers is that they work by supervised learning, where the classifier is trained on data with a known output, and then used on data of the same kind, allowing it to use its knowledge from the training data to classify the new data. It is important that the classifier can generalize and can sort data it has never encountered before, based on which sub-group it is most alike.

5.2. ARTIFICIAL NEURAL NETWORK

An artificial neural network is a classifier modeled after how the human brain works, which is very different from how one usually writes computer code.

A human brain contains an enormous amount of nerve cells, neurons. Each of these cells are connected to many other similar cells, creating a very complex network of signal transmission. Each cell collects inputs from all other neural cells it is connected to, and if it reaches a certain threshold, it signals to all the cells it is connected to.

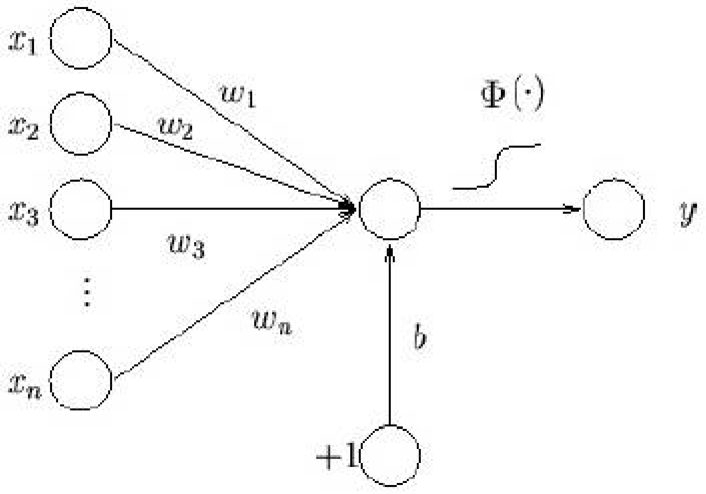


Figure 11: A graphical representation of a simple perceptron. Here y is the output signal, φ is the activation function, n is the number of connections to the perceptron, wi is the weight associated with the ith connection and x$_i$ is the value of the ith connection. The b in the figure represents the threshold.[46].

When writing an ANN, this is mimicked by using a "perceptron" as the basic unit instead of the neuron1. The perceptron can take several weighted inputs and summarize them, and if the combined input exceeds a threshold it will activate and send an output. Which output it sends is determined by the activation function and is often chosen to be between 0 and 1 or -1 and 1. Since the derivative of the activation function is often used in the training of the network, it is convenient if the derivative can be expressed in terms of the original function value, as few additional computations are needed to calculate the derivative in this case. The equation for a perceptron can be written as

$$y = \Phi\left(\sum_{i=1}^{n} w_i x_i + b\right) \quad \text{------------------------------5.1}$$

where y is the output signal, φ is the activation function, n is the number of connections to the perceptron, wi is the weight associated with the ith connection and x$_i$ is the value of the ith connection. b represents the threshold. A graphical representation can be found in figure 11. The threshold b is a neuron with a constant value of -1. By allowing the network to modify the weight associated with b, a dynamic threshold for when the perceptron activates is achieved.

This is a very simple design, and its strength can be shown when several perceptrons are combined and work together. The perceptrons are often organized in layers, whereeach layer takes input from the previous, applies weights and then signals to the next layer if appropriate. For a graphical representation, see figure 12.

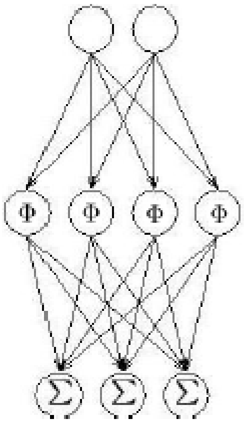


Figure 12: A graphical representation of an artificial neural network with one hidden layer. [46]

As mentioned in 5.1, a classifier must be able to learn from examples and adapt.

In an ANN, this is achieved by updating the weights associated with the connections between the layers. There are several ways of doing this, and most involve initializing the weights and fed the network an example. The error made by the network at the output is then calculated and feed backwards through a process called "back-propagation". This process is then used to update the weights, and by repeated use of this process, the network can learn to distinguish between several different classes. The exact equations involved vary from case to case, and the ones relevant to this project will be discussed in section 6.4.

To make the training more effcient, techniques such as momentum can be used. Momentum is used to and the right update step for the weights. If the step is too small the network will take too long to converge, while if it is to large the network might never converge and begin to oscillate instead. When using momentum, the step size is calculated dynamicaly during the run, on the basis that a weight which is changed often in the same direction most likely should be big, and can be changed faster. Another important techniqe is weight decay, which scales down all weights after every itteration. This means that large weights have to constantly be maintained to stay large, and avoids weights growing improportionally large.

One problem with the ANN approach is over-fitting of the data, which happens when the classifier becomes to good at recognizing the training examples, at the expense of not being able to recognize a general input. This can be avoided by cross-validation, where the network is trained on one set of data, and then evaluated on a separate one. When the error starts rising in the validation set, the network might be over-_tted. If previous networks are saved, the network can then be rolled back to the one which gave the smallest error. [47]

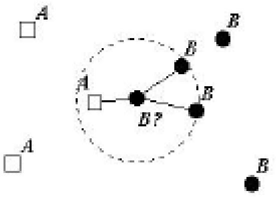


Figure 13: KNN classifier with K = 3 compares the new sample with the 3 closest ones. The classifier classifies the new set as B, since 2=3 of the closest sets are labeled B. If k had been equal to 1, the set would have been classified as A.

5.3. ARTIFICIAL NEURAL NETWORK (ANN) APPLICATIONS

ANN is fast and accurate because after the training process is completed, optimization and time-consuming calculations are no longer needed. So, the network outputs are predicted directly for the provided inputs based on what it has learned to predict for a specific system. There are many ANN types that are used for various applications such as engineering, weather and flood forecasting, business, and medicine because of their power and ability to generalize any practical problem (Coit et al., 1998; Twomey et al., 1998).

Generally, ANN applications fall into the categories of data clustering, classification, or regression. Data clustering creates relationships between fed inputs and separates them into different clusters based on their similarities. In classification, inputs are assigned to their class among different classes. Data regression means creating a curve that passes and fits between training data sets. The data regression type is normally used to predict and solve DHM applications. The main regression types of ANN are FFNN and RBNN, both of which have other subtypes under different names. The ANN detailed architecture and common types will be described in CHAPTER 2. Variables that could be used as input parameters in DHM applications include, but are not limited to, human anthropometry, the task to be performed, load existence, position (sitting or standing), joint ranges of motion (ROMs), and model DOFs.

Researchers incorporate ANN when they want to save time or cost in system development, or when they are unable to represent the system with a mathematical algorithm. For example, ANN was used to find the Cobb angle, which indicates scoliosis severity, by selecting the optimal set of input torso indices (Jaremko et al., 2002). The Cobb angle (ANN output) was calculated with accepted accuracy. Tani et al. (2008) trained a recurrent neural network (a type of ANN) on a humanoid robot to learn to manipulate objects. The results showed that the network can afford both generalization and context dependency in generating skilled behaviors. In addition, ANN was used in linguistics by Collobert et al. (2008) for language processing predictions. For a given sentence (ANN input), they trained the network to predict part-of-speech tags, chunks, named entity tags, semantic roles, semantically similar words, and the likelihood that the sentence makes sense.

5.4. COMMON TYPES OF ARTIFICIAL NEURAL NETWORKS

Many types of ANN have been developed to be used for many applications. Even for the same type, there are ANNs that differ in transfer functions and training approaches. Thus, selecting the most appropriate ANN type for a specific problem is not trivial. In this section, we will talk about the two main types of ANN that are used specifically to solve regression problems, which are the type of DHM applications in this thesis. These ANN types will be presented in terms of their general architectures, advantages, disadvantages, and applications.

5.4.1. FEED-FORWARD NEURAL NETWORK (FFNN)

Feed-forward neural network (FFNN), which is shown in Figure 14, is one of the most common and first developed types of ANN. Inputs are included in the input layer, which is shown in the figure as a set of circles. The inputs enter the hidden layer by the neuron weights that are shown in the figure. The hidden neurons are represented as circles each inside with a sigmoidal transfer function. The output layer receives the outputs of the hidden layer neurons by another set of neuron weights. Inside each neuron in the output layer, there is linear transfer function, shown in the same figure, to provide the final results (outputs). Generally, the sigmoidal and linear transfer functions are used on the hidden and output layers, respectively, when the problem is a regression type.

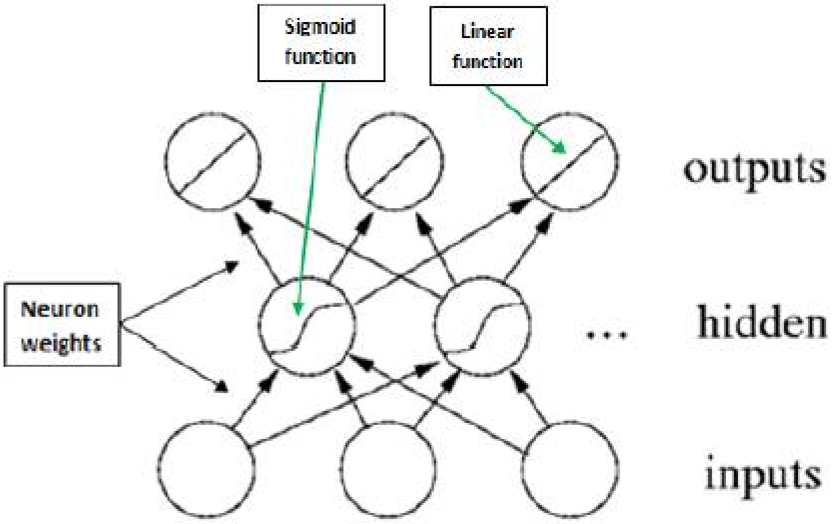


Figure 14: Feed forward neural network (FFNN).

FFNN is widely used because of its use in applications in both classifications and regression problems. The advantages of using FFNN are as follows:

1. Generalizing system prediction at any input or extrapolating off-grid training space. After the network is trained, it will be able to predict any new input, even those out of the training limits.
2. Working well for many applications, especially curve fitting of the time series data (i.e., data that come in different times and values).

FFNN, however, has some limitations that constrain using it for some applications. These limitations include the following:

1. It could be highly inaccurate because of local minima solution that comes from optimization. Usually, FFNN has more neurons in its hidden layer than other types of ANN. So, a local optimization solution is more likely to occur in FFNN.

2. It experiences training time and memory issues during the training process because it has more neurons to be optimized.

Therefore, these limitations exclude FFNN as an option in some applications when the number of the training cases and/or inputs and outputs are large. It is also excluded when high accuracy is required for system performance.

5.4.2. RADIAL-BASIS NEURAL NETWORK (RBNN)

Figure 15 shows the radial basis neural network (RBNN), which is another type of ANN that is widely used in various applications. Besides the input and output vectors, the network consists of one hidden layer and one outputs layer. Because RBNN provides the foundation for this work, we provide additional details regarding its structure.

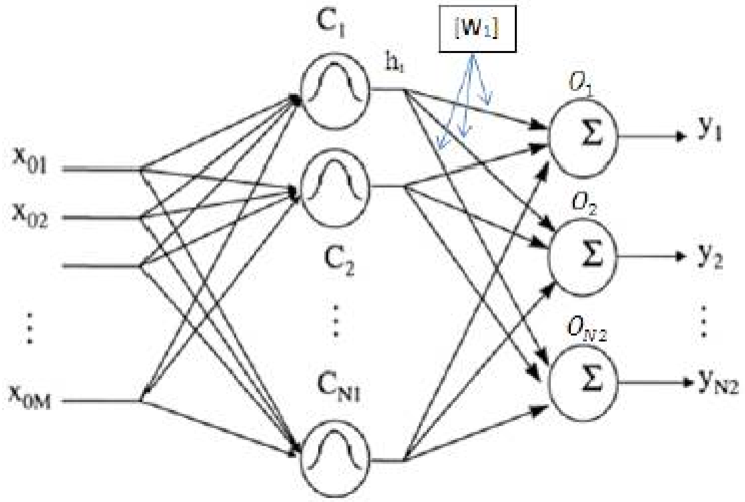


Figure 15: RBNN with M-dimensional input and N-dimensional outputs.

In the figure, $x = [x_{01}, x_{02},.......,x_{0M}]$ represents inputs of the network, [$C_{1,}$, C_2,.......,C_{N1}]are the neurons of the hidden layer, [W_1] is the vector of weights at the first neuron in the hidden layer (called line weights), and [y_1, y_2,.....,y_{N2}] represent the network's outputs.

In the figure, , x = [x_{01}, x_{02},.......,x_{0M}] provides the input for each neuron in the hidden layer, labeled C1 in the figure. In this case, the neurons are an essential radial basis function, hence the name radial basis neural network. All of the neurons collectively constitute the hidden layer. The hidden layer has N_1 neurons [$C_{1,}$, C_2,.......,C_{N1}]. Inside each hidden neuron C_i(**x**), there is a radial transfer function that produces output hi. The output hi is multiplied with weight vector W_i to produce hidden output vector $\boldsymbol{A_i}$. The dimension of the weight matrix, as shown in Equation 5.1 , and hidden output matrix ***A*** is *N2xN1.* Each row of ***W*** and ***A*** is referred to as a weight and hidden output vector associated with a corresponding neuron. The output layer has a number of neurons, labeled O1 in the figure, equal to number of outputs [y_1, y_2,.....,y_{N2}]. Inside each output neuron O_i($\mathbf{A}_i$), the output is calculated by taking the sum of the received lines $\boldsymbol{A}_i$, which represents a column of matrix ***A***. A full description of this network and its functionality are provided by Buhmann et al. (2003).

$$\boldsymbol{W} = \begin{bmatrix} w_{11}\ w_{12} \cdots w_{1N2} \\ w_{21}\ w_{22} \cdots w_{2N2} \\ \cdots \cdots \cdots \cdots \cdots \cdots .. \\ w_{N11}\ w_{N12} \cdots w_{N1N2} \end{bmatrix} \qquad [5.1]$$

$$\boldsymbol{h} = [h_1\ h_2 \ldots\ldots h_{N1}] \qquad [5.2]$$

$$\boldsymbol{A} = \boldsymbol{h^T} \cdot \boldsymbol{W} \qquad [5.3]$$

$$y_i = \sum_{k=1}^{N1} A_{ki} \qquad [5.4]$$

RBNN is trained by solving the optimization problem in Equation 5.5

$$\text{Find:} \qquad \boldsymbol{W_{N1XN2}} \qquad [5.5]$$

$$\text{To min:} \qquad MSE = \sum_i^N (T_i - y_i)^2$$

In the above formula, T_i represents the i^{th} training output, □□ is the predicted output from the network. Note that y is a function of **W**, as shown in Equation 5.4. The training starts with the first iteration with one hidden neuron (N1=1). Then, N1 is incremented by 1 each time before the next iteration. The optimization stops once the MSE equals a small value (almost zero).

Like FFNN, RBNN is used for all applications in both classification and regression problems. In this thesis, the RBNN is specifically used because of the following superior advantages:

1. It provides highly accurate results within the limits of the training space (i.e., inside the domain of the training values).

2. There are no local minima problems. The network does not optimize to local minimum solutions because the number of hidden neurons is optimized automatically in the training process. Thus, the optimal solution is obtained in terms of the number of neurons and the network weight matrix ***W***.

3. There are no computational time and computer memory problems, especially when there are a large number of input/output training sets, because the network does not have a large number of neurons and weights. The weight values to be optimized exist only on the output side of the hidden layer, while FFNN has weights in both sides.

4. It was found by experience that RBNN is the best type of ANN for highg dimensional regression models.

Although RBNN has powerful prediction capabilities, it has some expected limitations, as follows:

1. The network parameter (Gaussian width) is determined heuristically, which could produce poor results.

2. It cannot predict points that are out of training grid space. The network cannot provide accurate outputs when the input is outside the range of training data (i.e., no extrapolation).

5.4.3. MULTI-LAYER PERCEPTRON NEURAL NETWORK

Multilayer Perceptrons are a powerful tool used to build predictive models from a set of input data containing a given number of features and as result predict one or more target variables. The topology of a MLP is simple and straightforward, see figure 16. The network is divided into layers, which comes in three flavours: input layer, hidden layers and output layers. For each feature in the input data a separate node is created in the input layer. One or more hidden layers that can contain different number of nodes each and finally an output layer with one or more nodes follow this layer. In this paper a single output node was used to produce the estimation of the apartment price (regression). When the MLP model is used for classification a ‘Soft Max’ is created which consists of one node for each expected class.

Each node in the lower layer is connected to all nodes in the layer above, thus forming a complete bipartite graph. A weight is associated with the connection, for example w_{hi} in figure 16. The output of a node in the network is calculated with an

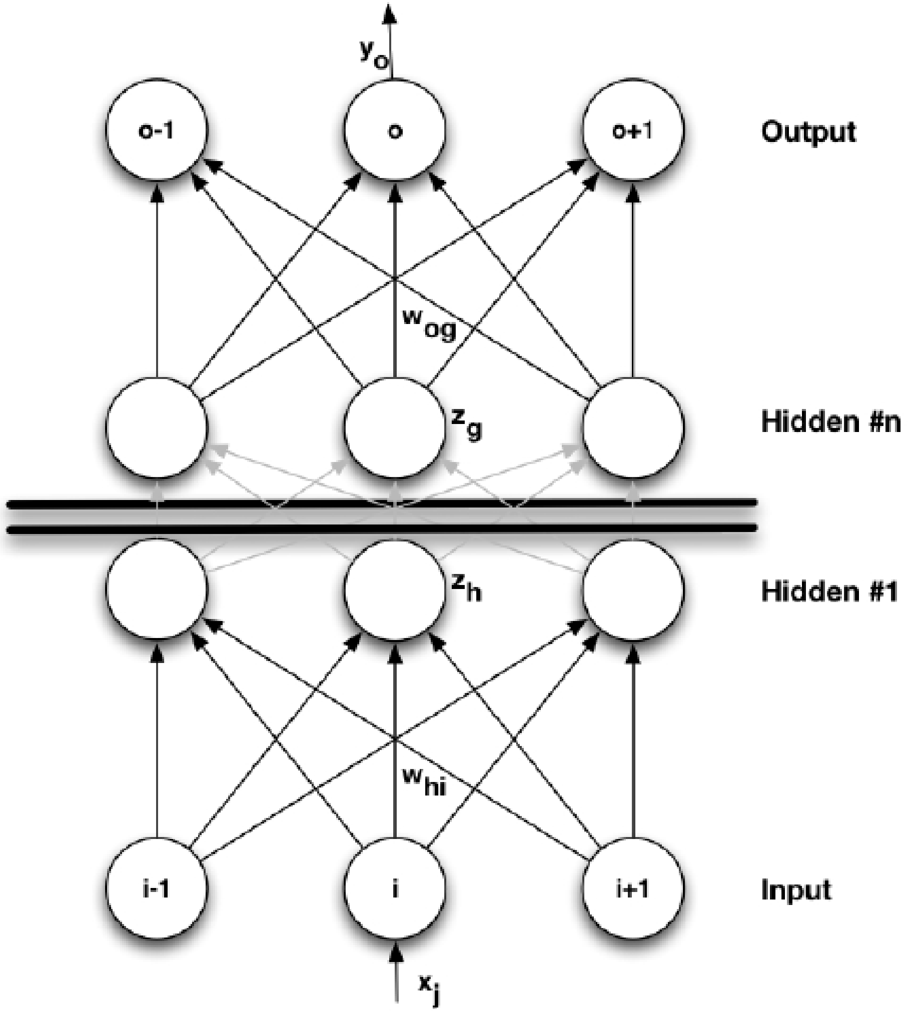


Figure 16: Layout of *Multilayer Perceptron*.

activation function which input is the weighted sum of the incoming connections. Activation functions are covered in section 5.4.3.1. An error function is used to calculate the difference between the output from the network and the target (desire) value, this is typically the mean square error function $E = 1/2\ (t - y)2$ but other functions are also used. In this paper the supervised training algorithm called backpropagation was used to create the statistical model. This algorithm can briefly be described as follows:

1. Sample from the training set is presented (input data to input nodes).
2. Inputs are propagated forward in the network by calculating output values for the nodes in each layer by applying the activation function from input nodes towards output node. *Forward propagation*
3. Output error is calculated by the error function $E = 1/2\ (t - y)2$. Here t is the target value and y is the output of the network.
4. Calculate the gradient, momentum ($M(t) = M(t - 1) * \boldsymbol{\lambda} - gradient$) and update weights ($W(t) = \alpha * M(t)$), here $\boldsymbol{\lambda}$ = velocity decay, α = learning rate. *Backward propagation*

The algorithm described above is applied on the whole test data set and repeated for the desired number of iterations. At this point all weights are adjusted to represent a good model of the problem. Initialization of the weights is discussed in section 5.4.3.2. Three different regimes of weight updates are often used:

- *Online*. Weights are updated after every sample in the test dataset.
- *Batch*. Weights are updated after passing all training data.
- *Mini-batch*. Divide the training data set into chunks of equal size and update the weights after passing a chunk.

5.4.3.1. ACTIVATION FUNCTION

In the early work of this thesis the problem was studied using the machine learning tool Weka to find the best path for the work at hand and what restrictions to be aware off. Ample efforts were used trying out different activation functions, assorted features sets and different parameter settings. It was obvious early on in this work that it was not feasible to use linear activation functions and that the sigmoid activation function was unable to produce good models using the features available in this study. To overcome this problem Weka was enriched with a hyperbolic tangent activation function in order to study its behaviour on the feature set at hand.

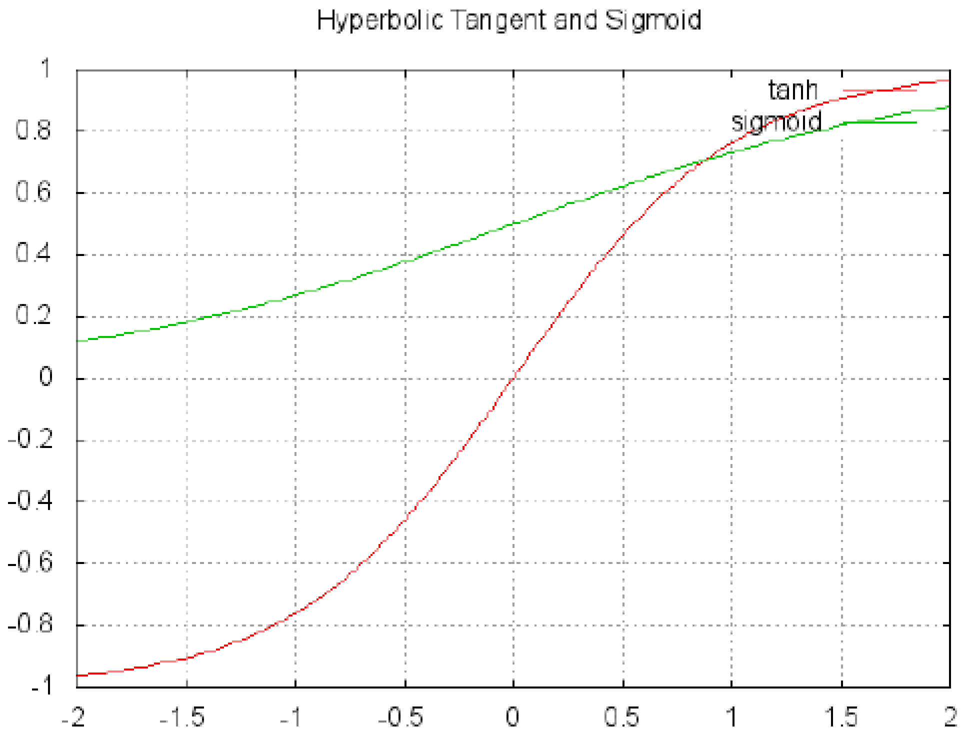


Figure 17: Activation function

5.4.3.2. WEIGHT AND BAIS INITIALIZATION

The most commonly used weight initialization scheme used is often referred to as regular initialization presented in equation 5.6 below.

$$W_{ij} \sim U[-\frac{1}{\sqrt{n_{i-1}}}, \frac{1}{\sqrt{n_{i-1}}}] \qquad [5.6]$$

Inadequate initialization of the weight can lead to saturation problems for the weights and give negative effects upon the gradient descent algorithm used to build the model. This can render an inexpert model that is unable to do adequate prediction.

5.4.3.3. WEIGHT UPDATE REGIME

Multilayer perceptron models can with advantage be built with use of the Backpropagation algorithm and have Gradient Descent as one of its corner stones. Choosing a good regime of weight updating is therefore crucial. It is more rewarding to follow small but consistent gradients when updating the weights than bigger and more inconsistent ones. In this paper we used two mechanisms to refine the process of updating the weight: learning rate and momentum. The concept of adding learning rate can be viewed, as a way to control how fast the weights should be learned in an update. For data sets with redundant data the learning rate can be low though too low learning rate will slow down the learning considerable, too high rate can make the learning overshoot. It is often favourable to keep the learning rate high in the beginning and to turn it down further along in the update process.

The method of using momentum stems from the idea of adding a momentum to the current gradient in the gradient descent algorithm rather than following steepest descent. Adding a momentum based on the previous weight updates to the current gradient makes it keep going in the previous direction, a momentum, see equation 5.7.

$$\mathbf{v}_t = \alpha \mathbf{v}_{t-1} - \epsilon \frac{\partial E_t}{\partial \mathbf{w}_t} \quad [5.7]$$

$$\Delta \mathbf{w}_t = \mathbf{v}_t \quad [5.8]$$

$$\Delta \mathbf{w}_t = \alpha \Delta \mathbf{w}_{t-1} - \epsilon \frac{\partial E_t}{\partial \mathbf{w}_t} \quad [5.9]$$

Weight update can be expressed in terms of the velocity, see equation 5.8. Expressing the update in terms of previous weight update gives the equation 5.9. This combined with the learning rate gives the final update function $\boldsymbol{\lambda \Delta w}_t$ where $\boldsymbol{\lambda}$ is the learning rate and α the momentum multiplier.

5.5. THE MATHEMATICS OF BACKPROPAGATION

The model consists of L feature variables, this is the same number as input neurons, figure 18 shows the configuration of the network. Each record in the training data is comprised of the feature variables x = {x$_{(1)}$, x$_{(2)}$,…..,x$_{(L)}$} and a target variable y. The training set consists of M tuples as follows:

T ={(x$^{(1)}$,y$^{(1)}$),(x$^{(2)}$,y$^{(2)}$),……..,(x$^{(M)}$,y$^{(M)}$)}

5.5.1. LAYOUT OF THE NEURAL NETWORK

This paper will mainly cover multilayer perceptrons with input nodes, one or two hidden layers and a regression output node. Note that only regression output will be used so the output unit is linear and no Softmax will be included. Let I denote the number of output neurons and H,G the number of hidden units, see figure 18. Finally the number of input neurons is given by L.

5.5.2. ERROR FUNCTION

The error function is used to measure the error between the actual value and the prediction. Define the error function E (equation 5.10) as the sum of the squared difference between the expected and calculated output , *n* ∊ *trainingset*. Note that the error function for the regression case is different from the function
used with classification.

$$E = \frac{1}{2}\sum_{n}(t^{(n)} - y^{(n)})^2 \qquad [5.10]$$

Taking the derivative of E (equation 5.10) with respect to the weights gives us the following function:

$$\frac{\partial E}{\partial w_i} = \frac{1}{2}\sum_{n}\frac{\partial y^{(n)}}{\partial w_i}\frac{\partial E}{\partial y^{(n)}} = -\sum_{n}\frac{\partial y^{(n)}}{\partial w_i}(t^{(n)} - y^{(n)}) \qquad [5.11]$$

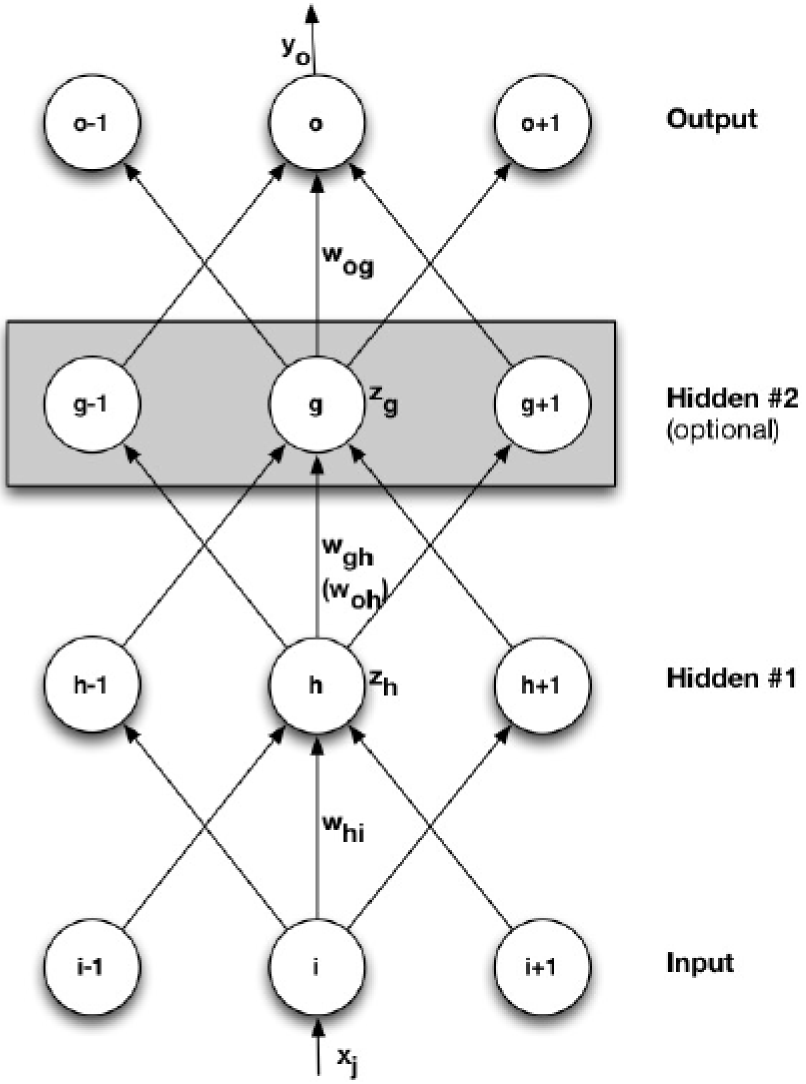


Figure 18: Layout of *neural network*. Note that the hidden layer in the lower part of the figure are optional, both network with one and two hidden layers will be discussed.

We can now form the batch delta rule Δw*i* as

$$\Delta w_i = -\epsilon \frac{\partial E}{\partial w_i} = \sum_n \epsilon \frac{\partial y^{(n)}}{\partial w_i}(t^{(n)} - y^{(n)}) \qquad [5.12]$$

5.5.3. ACTIVATION FUNCTIONS IN THE NODES

In this paper we are using sigmoid activation function. In the hidden nodes we use a sigmoid or activation function. The output regression node is linear. Here we present the equations for the activation functions used and its derivative.

5.5.3.1. LOGISTIC NEURON (SIGMOID)

Define the activation function for the logistic neuron as

$$y = \frac{1}{1 + e^{-z}} \qquad [5.13]$$

Where

$$z = b + \sum_i x_i w_i \qquad [5.14]$$

The derivative of *y* is $\frac{\partial y}{\partial z}$ = *y*(1 - *y*)

Partial derivatives for z are $\partial z/\partial w_i = x_i$ and $\partial z/\partial x_i = w_i$. Now can the partial derivative of *y* with respect to w_i be defined

$$\frac{\partial y}{\partial w_i} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i} = y(1 - y)x_i \qquad [5.15]$$

5.5.4. FINDING THE GRADIENTS FOR THE ERROR FUNCTION

In this section the gradient for the error function both for multilayer perceptrons with singe layer of hidden units is defined. The notation is based on the network configuration shown in figure 18.

5.5.4.1. SINGLE HIDDEN LAYER WITH SIGMOID ACTIVATION FUNCTION

We need to find $\frac{\partial E}{\partial woh}$ and $\frac{\partial E}{\partial whi}$ in order to perform the calculations required by the back propagation algorithm. As before ε is the learning of the gradient decent. We use Δw_{oh} and Δw_{hi} to update the weights w_{oh} and w_{hi} respectively. N is the number of observations in a minibatch and $n \in minibatch$.

The partial derivatives of the linear activation function are

$$\frac{\partial y}{\partial w_i} = x_i \qquad [5.16]$$

And

$$\frac{\partial y}{\partial x_i} = w_i \qquad [5.17]$$

For the first weight between the output node and the hidden unit we need to calculate Δw_{oh}.

$$\Delta w_{oh} = \epsilon \frac{\partial E}{\partial w_{oh}} = \epsilon \frac{\partial E}{\partial y_o^{(n)}} \frac{\partial y_o^{(n)}}{\partial w_{oh}} \qquad [5.18]$$

From equation (5.11) we get $\frac{\partial E}{\partial y}$ and from equation (5.16) we get $\frac{\partial y}{\partial woh}$. This gives us the solution for equation (5.18) as

$$\Delta w_{oh} = \epsilon \sum_n (t^{(n)o} - y_o^{(n)}) z_h^{(n)} \qquad [5.19]$$

The delta for the weights between the input node and the hidden layer is given by Δw_{hi}.

$$\Delta w_{hi} = \epsilon \frac{\partial E}{\partial w_{hi}} = \epsilon \frac{\partial E}{\partial y_o^{(n)}} \frac{\partial y_o^{(n)}}{\partial z_h^{(n)}} \frac{\partial z_h^{(n)}}{\partial w_{hi}} \qquad [5.20]$$

From equation (5.11) we get $\frac{\partial E}{\partial y}$ and from equation (5.17) we get $\frac{\partial y}{\partial zh}$. The final part $\frac{\partial zh}{\partial whi}$ can we obtain from equation (5.15). This gives us a solution for equation (5.20) as

$$\Delta w_{hi} = \epsilon \frac{\partial E}{\partial y_o^{(n)}} \frac{\partial y_o^{(n)}}{\partial z_h^{(n)}} \frac{\partial z_h^{(n)}}{\partial w_{hi}} =$$

$$\epsilon \sum_n \sum_o (t_o^{(n)} - y_o^{(n)}) w_{oh} \, z_h^{(n)} (1 - z_h^{(n)}) x_i^{(n)} \qquad [5.21]$$

CHAPTER 6

PROPOSED SYSTEM

6.1. PROPOSED SYSTEM VISION

The proposed system will be smart enough to know the context, mood of the user and status of the user and to recommend to him what he wants. Below are some scenarios of how this system will work:

- The system may recommend a specific user a movie depending on his location gotten from the gps of his mobile, his past history, the genre of movie he likes, weekday or weekend. For example, If user's mood is negative he may like classical slow songs and if its weekend and user's mood is positive then he may like rock songs.
- The same system may recommend to the same user few songs depending on his location gotten from the gps of his mobile, his past history, genre of songs he likes depending on his mood and time.

6.2. PROPOSED SYSTEM DESIGN

In this section, I present the general design of the proposed multi-type context aware recommender system. Including the design of the main block in the system which is the Context Aware Management System (CAMS). Inside this block, there will be a design for the context reasoning block using a neural network. The results of training and testing this network will be introduced in the next section.

A. General System Design

The system generally has two phases, the situation and recommender type phase, and the items recommender phase. The proposed system design is shown on figure 8.

- **Situation and Recommender Type Phase:** the main function of this phase is to determine whether a situation is proper to push a recommendation or not, and to determine what type of recommendation to push from a predefined set of recommendation types. This phase is encapsulating a design of a CAMS relying on the context lifecycle mentioned in [9]. The CAMS consists of four stages: the first stage is the data acquisition, which retrieves the raw data from the IoT. The second stage is the data modeling that models (represents) the raw data to be understood, like in key-value modeling, markup scheme, graphical modeling, and object-based modeling. The third stage is the context reasoning, which is the method of deducing new knowledge based on the available context. The fourth stage is the scoring algorithm, which generates an output score based on the output of the context reasoning block. In our design the reasoning phase is implemented using a neural network that generates different scores for the recommendation types based on the context, then the maximum of these scores is considered; i.e. the scoring algorithm takes the maximum of the input scores. If this maximum score is less than a pre-determined threshold, then the next phase will not be triggered. But if this score is greater than the threshold, then the next phase in which a specific items belonging to the type that got the maximum score will be triggered.

- **Item Recommender Phase:** This phase is triggered from the previous phase. This is a traditional context aware recommender system based on collaborative filtering. As in [11], new scores are calculated for the items, then the items whose scores are above a threshold will be displayed to the user ranked from max to min. These scores corresponds to the predicted items ratings of a standard collaborative filtering. The threshold with which the items scores is compared is adaptive and changes according to the user feedback which is given by liking and disliking the recommended items.

B. Context Reasoning Block Design

In order to test the first phase of the system, we have designed the sub-blocks of the CAMS. The main design work was in the context reasoning block. This block is designed using a supervised feed-forward neural network which takes the modeled contexts as inputs and generates the scores of three pre-defined types of recommendations as outputs. This neural network consists of one input layer of 5 inputs, one hidden layer of 8 neurons, and one output layer of 3 outputs. More details about the design specifications, training, implementation, and testing of this neural network will be introduced in next section.

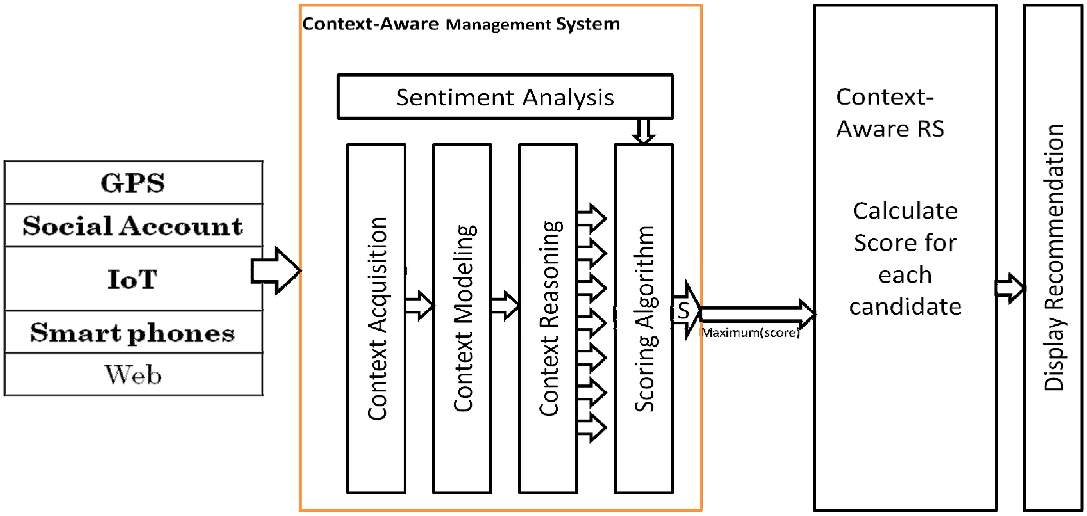


Figure 19: Proposed System Design

The Multiple Layer Perceptorn Neural Network is used in Context Aware Management System to classify the context of the user taken from Internet of things. The activation function used is Sigmoid in each perceptron. The context is given to the neural network from IoT virtually. The neural network takes five inputs from the IoT. The five inputs are:

- Time
- Latitude
- Longitude
- Mood
- Week

The time is considered to be taken from the smartphone but here its taken from the system (Lappy). Latitude and Longitude is considered to be taken from the smartphone’s GPS. Here it is taken using the Google API. Using Google Development App, the Access key is taken from google and the given address is converted into latitude and longitude.

Mood of the user is determined using the sentiment analysis of the recent tweet of the user. To extract the tweet from the user’s Twitter Timeline, Twitter Apps is created and by using the consumer key and consumer secret along with the access token , access token secret for the OAuth of Twitter App. I have used the usertimeline API of Twitter to extract the recent tweet of a specific user from his timeline using Twitter4j library. Next the sentiment analysis is done using the Standford Core NLP libraries. The Standard sentiment analysis is done on the tweet to determine the mood of user.

Twitter4J is an unofficial Java library for the Twitter API. With Twitter4J, you can easily integrate your Java application with the Twitter service. Twitter4J is an unofficial library.

Week has two values: 0 or 1. If its weekday then its value is 1 else 0. It is taken from the system date and changed accordingly.

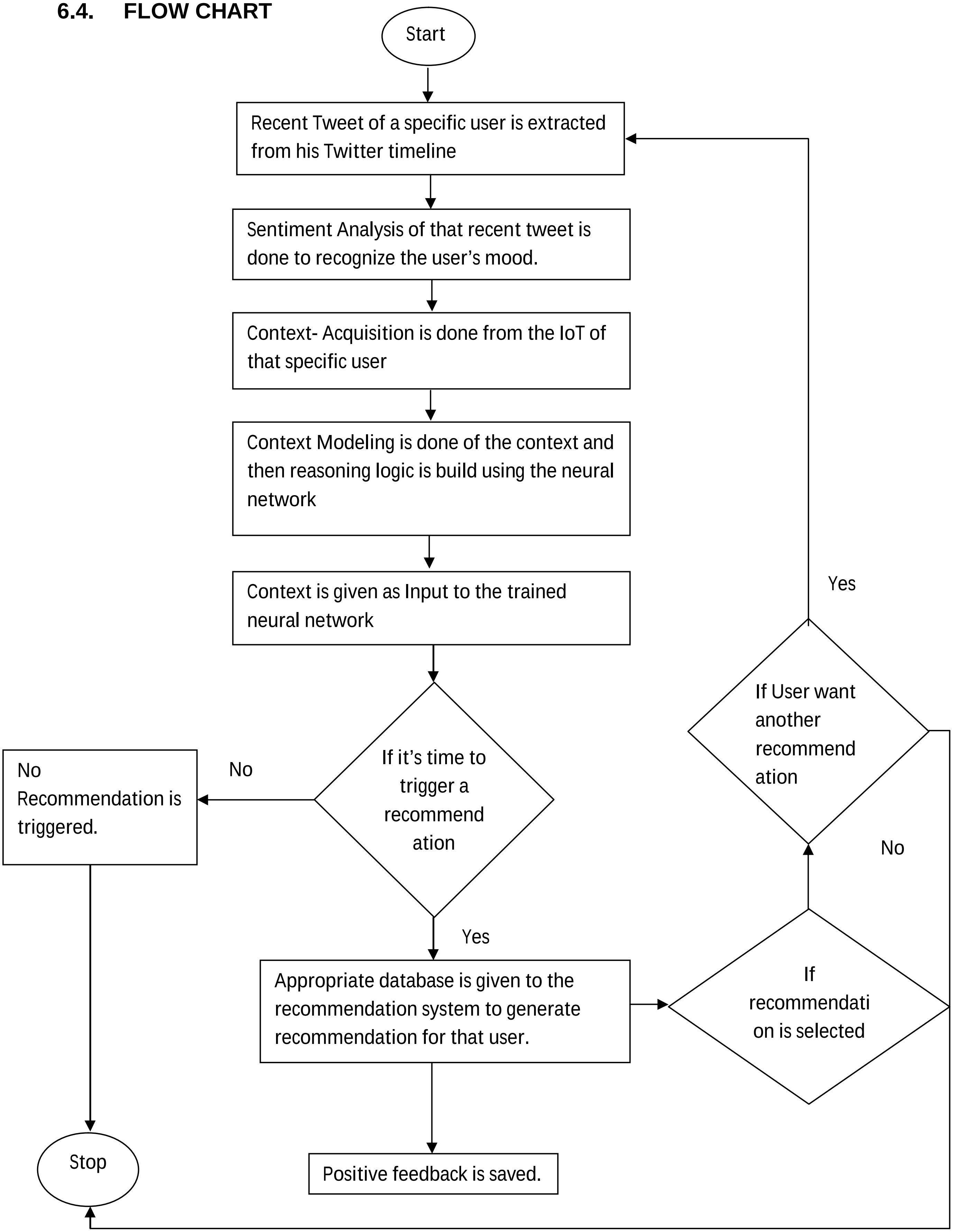
The output of the MLP neural network are: song, movie and none. The maximum value of all three is considered to triggered for the recommendation and the appropriate database is given to the system for recommendation. The trained neural network is shown in the next section.

Recommendation System user Collaborative Filtering Technique to produce the recommendation to the user. The standard collaborative filtering technique is used for the recommendation purpose.

6.3. ALGORITHM

1. First the recent tweet of a specific user is extracted form his timeline using the usertimeline API of Twitter with the help of Twitter4J libraries.
2. Now the Sentiment Analysis of the tweet is done to recognize the mood of the user using the standard sentiment analysis algorithm.
3. Now the context of user is taken from the IoT, this is called context acquisition.
4. Context Modeling is done of the data. Here key-value modeling is used to understand the data.
5. Now the Context Reasoning is done which is the method of deducing new knowledge based on the available context.
6. Now the input is given to the trained neural network and the score is calculated.
7. The maximum score output is triggered for the recommendation to the recommendation system.
8. If it’s time to trigger any of two: song and movie recommendation then the appropriate database is given as input to the recommender system.
9. The recommendation system uses the standard collaborative filtering for the recommendation purpose.
10. The top 10 recommendations are shown to the user.
11. If user like any recommendation then that song or movie get a positive feedback which helps in next time recommendation.

6.4. FLOW CHART


Start
Recent Tweet of a specific user is extracted from his Twitter timeline
Sentiment Analysis of that recent tweet is done to recognize the user's mood.
Context- Acquisition is done from the IoT of that specific user
Context Modeling is done of the context and then reasoning logic is build using the neural network
Context is given as Input to the trained neural network
If it's time to trigger a recommend ation
No
No Recommendation is triggered.
Yes
Appropriate database is given to the recommendation system to generate recommendation for that user.
If recommendati on is selected
If User want another recommend ation
Yes
No
Positive feedback is saved.
Stop

CHAPTER 7

METHODOLOGY AND TESTING

The previous design was implemented to trigger one of two recommender types, they are: Songs and Movies. For this purpose, the context acquisition block is assumed to collect data from the IoT, which include: GPS location, time, weekend or weekday, new songs or movies, rating or songs and movie and if it is holiday. Sentiment Analysis of the recent tweet is done using the neural network of Standford Core libraries. The mood of the user is calculated and also given as an input to the Neural Network to trigger one of two recommendation (songs and movies).

This data is modeled by key-value method. First the neural network is trained using only 100 random records and then tested using those records and the performance of the trained neural network is shown in figure 20.

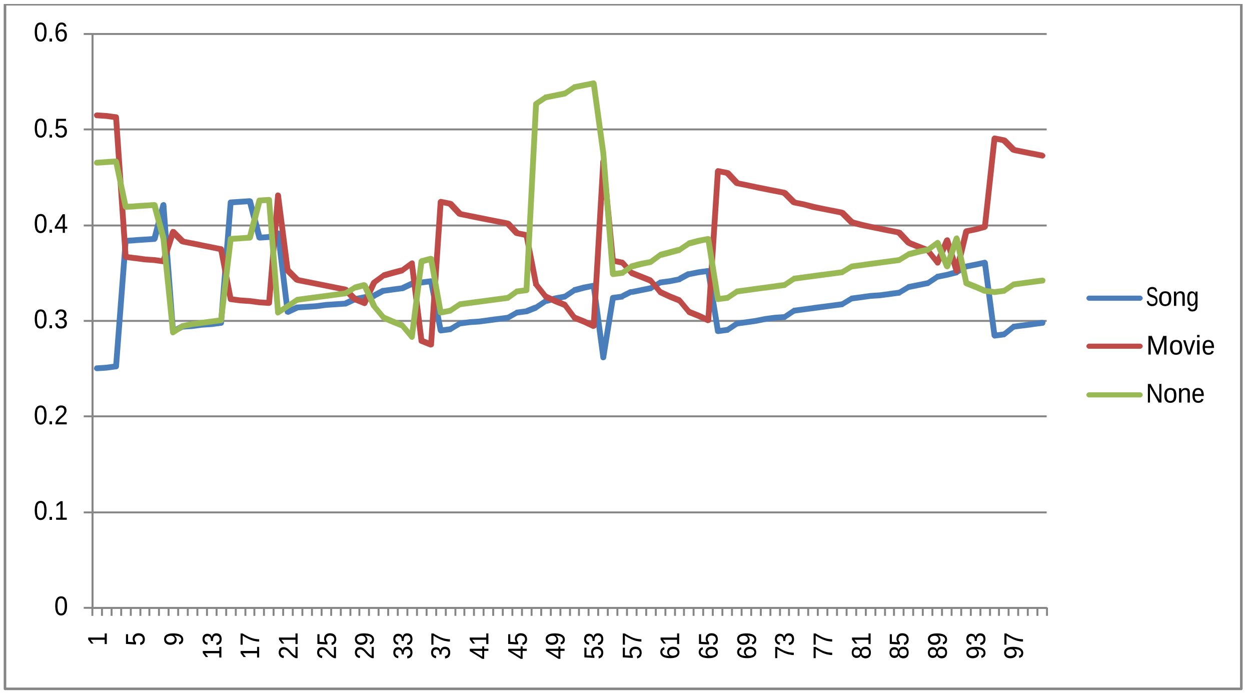


Figure 20: Actual Output of Trained Neural Network for 100 random records

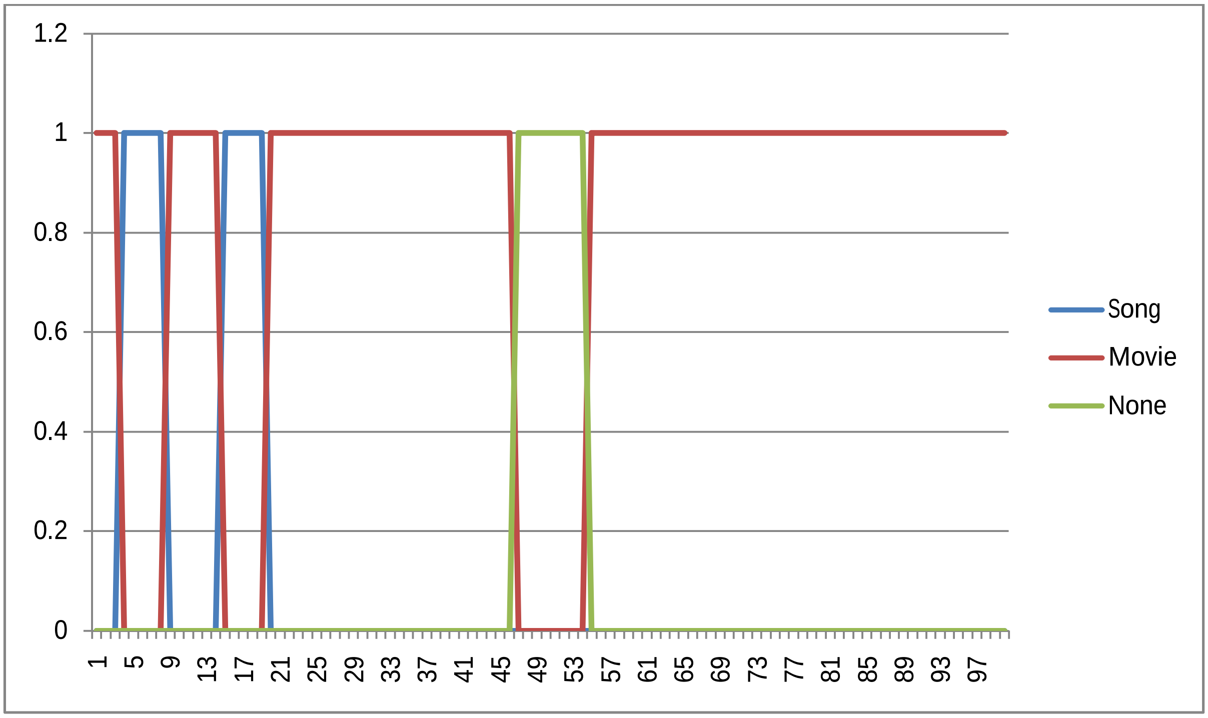


Figure 21: Expected Output of Trained Neural Network for 100 random records

The accuracy of this system is 91% on the 100 random records. As the accuracy is not good, the system is further trained using 500 random records and then tested using 500 random records. The result of the tested neural network is shown in figure 22.

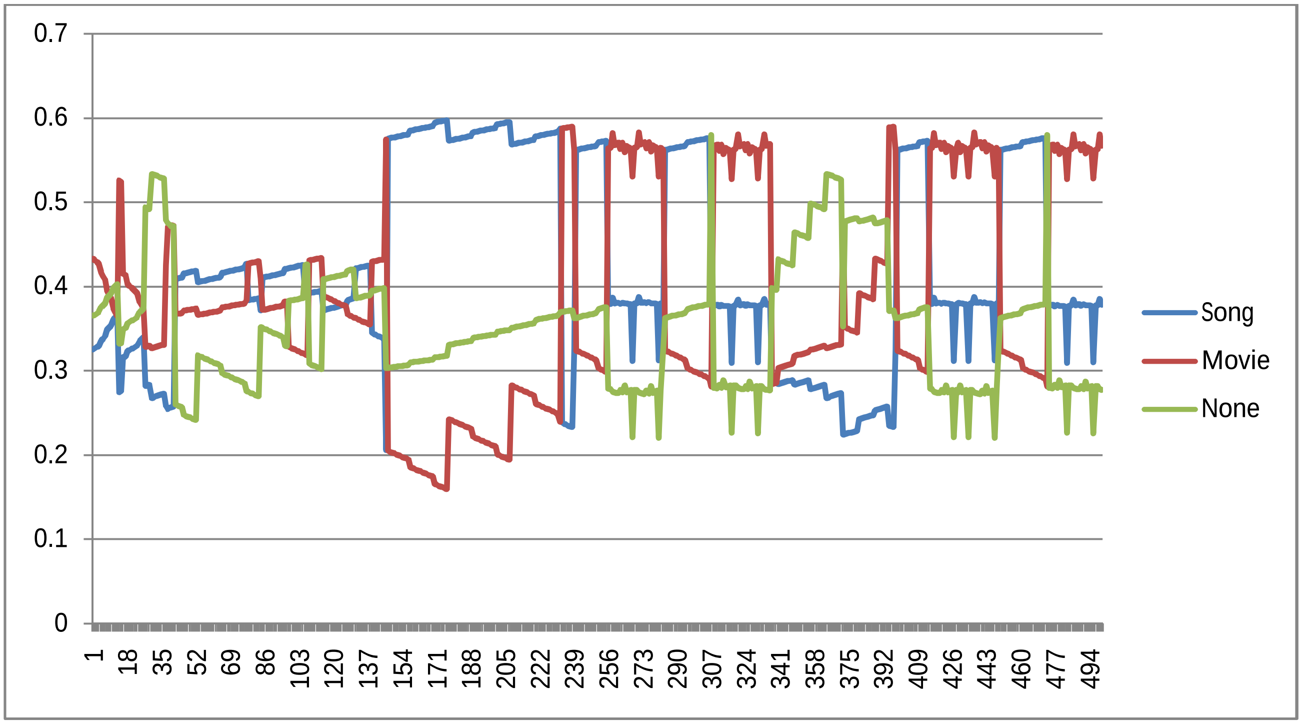


Figure 22: Actual Output of Trained Neural Network for 500 random records

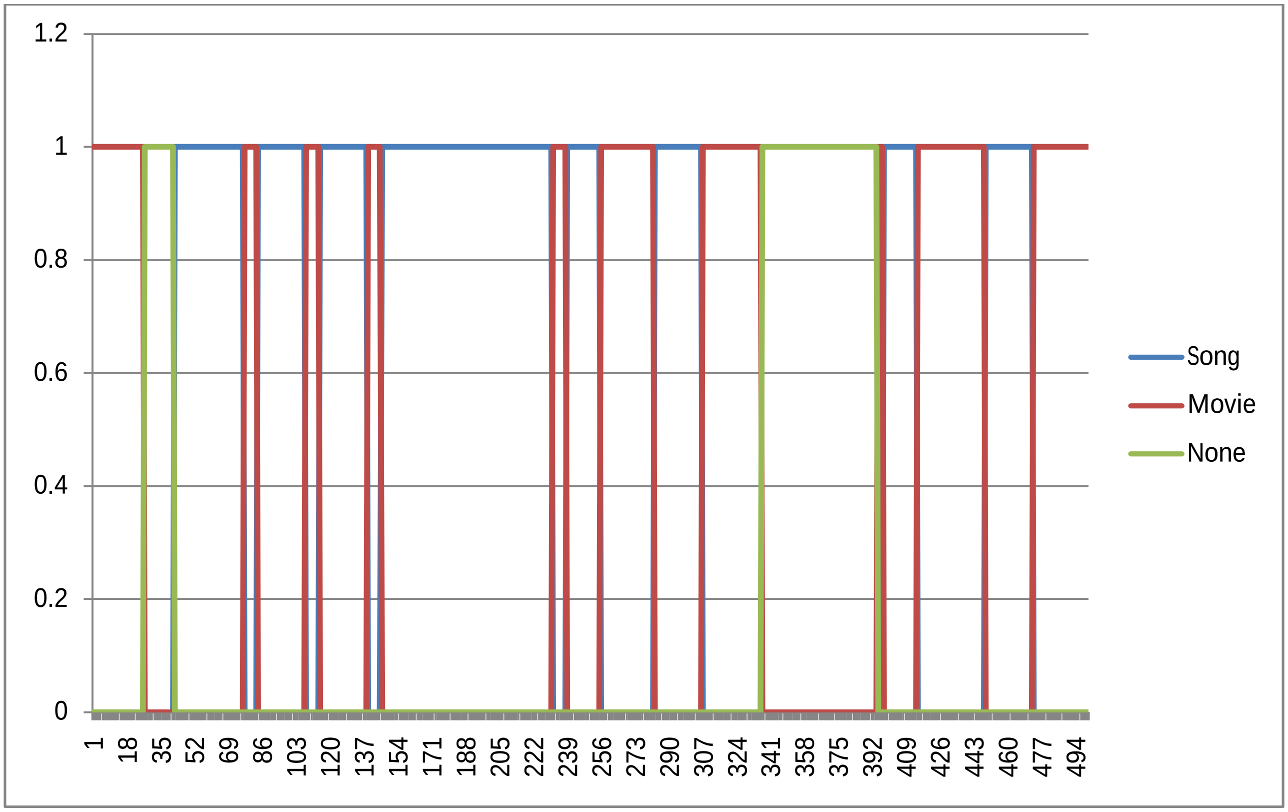


Figure 23: Expected Output of Trained Neural Network for 500 random records

This time system provide 98.80% accurate result. To improve the accuracy of this system it is trained using 100 random data values, in addition to the expected scores of each record for the three types, were created as training data. The scores in the training data were set to be between 0 and 1. Where the highest scores were given to the song, and the lowest scores were given to none. The training data records (excluding the locations latitude and longitude) were used to train the neural network shown in figure 24. The performance of the trained neural network is shown in figure 24.

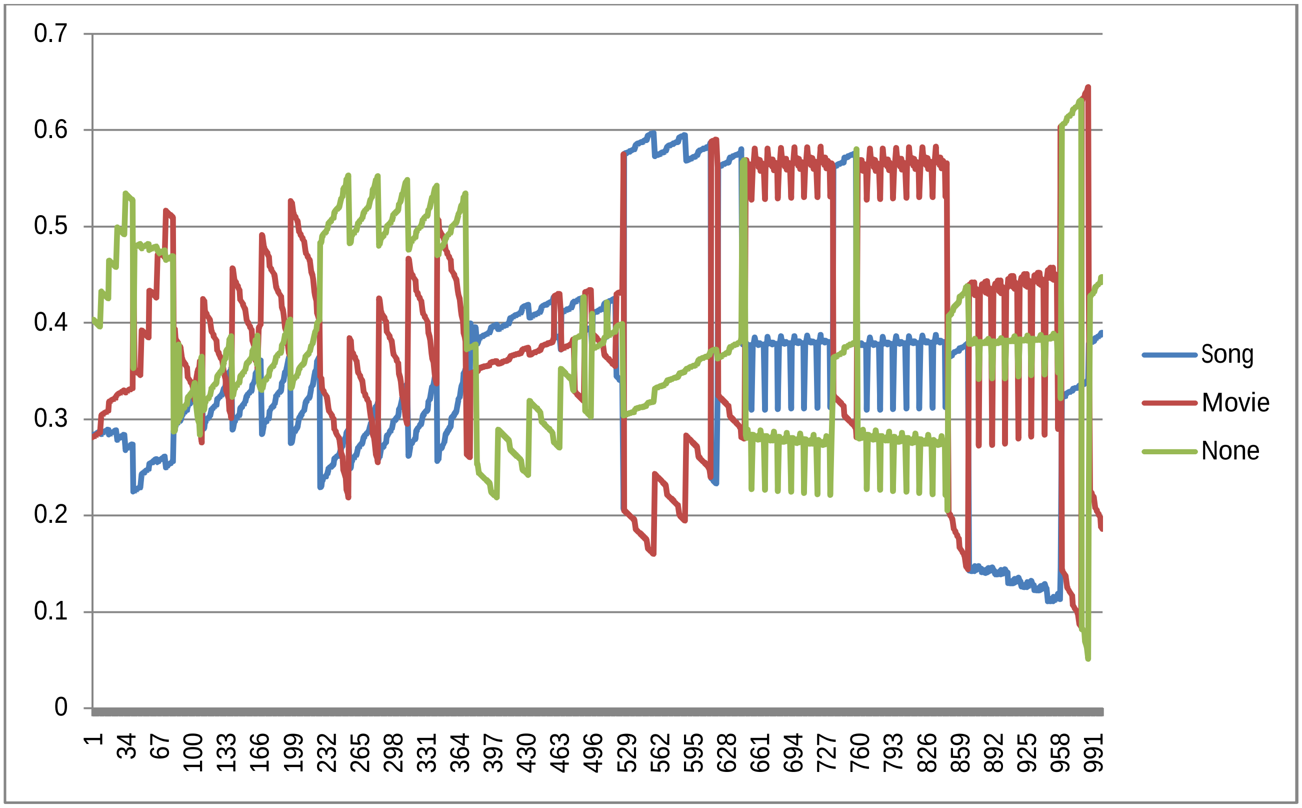


Figure 24: Actual output of the Trained Neural Network for 1000 Random Records

The trained neural network have three output. The maximum value of all the three output is triggered by the cams and the appropriate database is fetched for the recommendation system.

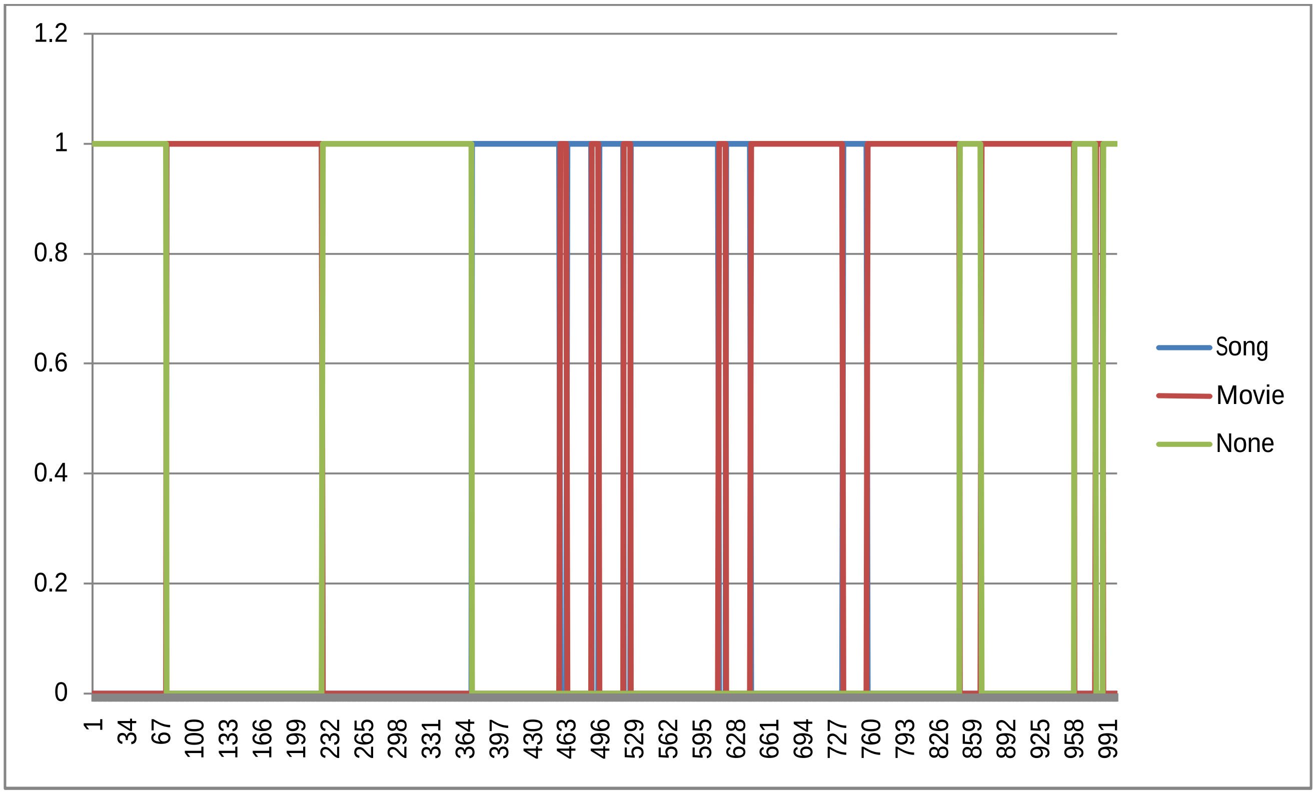
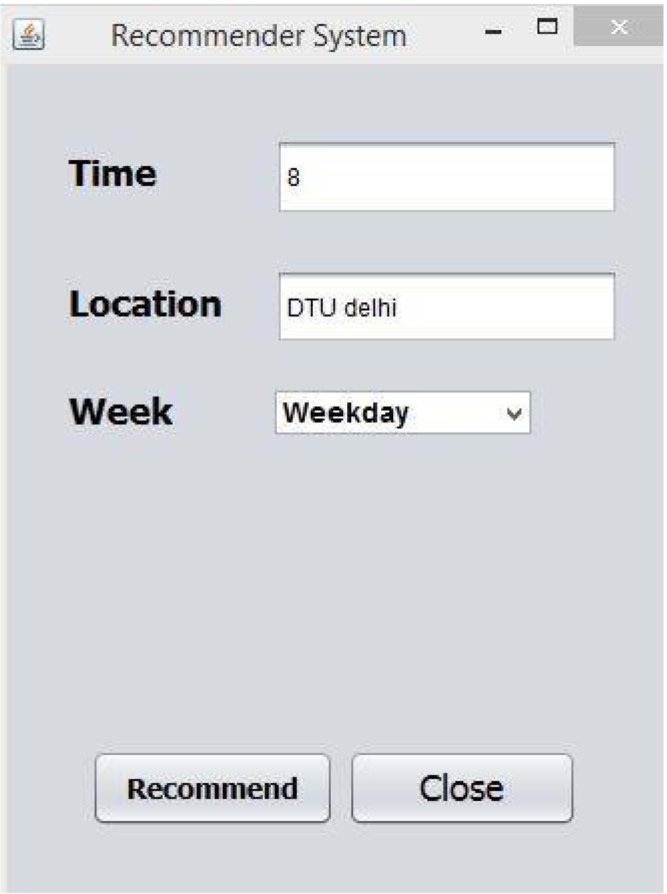


Figure 25: Expected Output of the Trained Neural Network for 1000 Random Records

The GUI of the System is created using the JFrame in Netbeans as shown in figure 26. The longitude and latitude of the place is calculated using the Google API. OAuth of the Google development apps is taken and the access key is created.

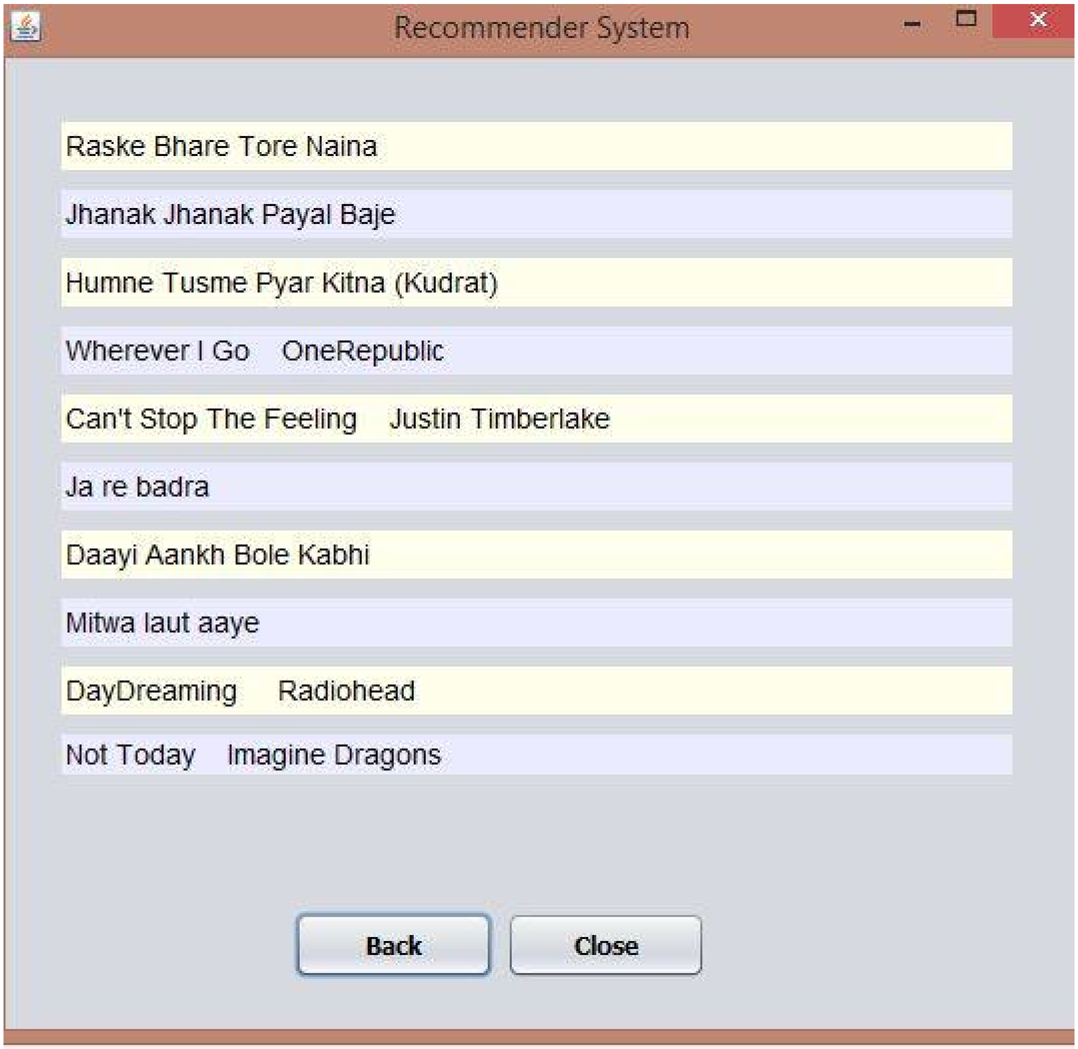


Time 8

Location DTU delhi

Week Weekday

Figure 26: GUI of the Recommender System



Raske Bhare Tore Naina

Jhanak Jhanak Payal Baje

Humne Tusme Pyar Kitna (Kudrat)

Wherever I Go   OneRepublic

Can't Stop The Feeling   Justin Timberlake

Ja re badra

Daayi Aankh Bole Kabhi

Mitwa laut aaye

DayDreaming   Radiohead

Not Today   Imagine Dragons

Figure 27: GUI of the recommended songs

CHAPTER 8

RESULT ANALYSIS

To be able to analyze the results thoroughly, a Java code that generate random input records was written. The generated random records were entered to the previously trained neural network; then it was run 1000 times to generate 1000 different recommendations types triggering.

First the system is tested using 100 random records and accuracy of the system is 91% found. To improve this accuracy the system is again trained using 500 random records and then tested using 500 random records and the accuracy is increased to 98.80%. Again the neural network is trained using 1000 random generated records and then tested using 1000 random records and the accuracy is 98.80%. As the system's accuracy quite acceptable enough hence the system required more data to train to get an accurate result. For new user it may recommend sometimes wrong things but as the system is used by the user it get better as the records increased and the accuracy of the system will also increase.

The result of the 1000 random records were 263 of 268 to trigger a song type recommendation. 452 of 458 to trigger a movie type recommendation and 273 of 274 not to push any recommendation. The result is shown in figure 28.

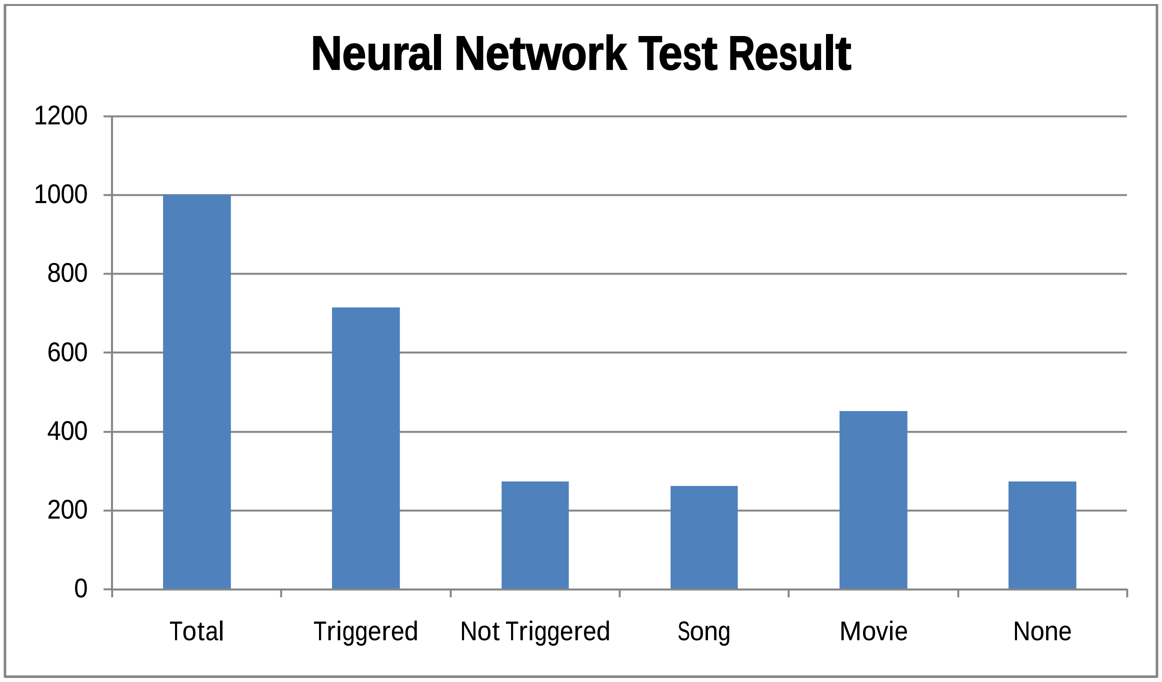


Figure 28: ANN Test Result

The accuracy average for the above types is 98.80 % which is considered a very satisfactory result for deciding whether to push or not to push a recommendation, and what type of recommendation to trigger.

CHAPTER 9

CONCLUSION AND FUTURE WORK

In this thesis, a design of a context aware recommender system has been proposed which recommend based on the mood of the user. In which different types of recommendations are provided to the user in the proper context. This system is promising in the environment of the IoT where much information about the user context will be available. It also uses the sentiment analysis of the tweets of a specific user for the calculation of the mood of that specific user. Using the neural network for the context reasoning provided more than 98% accuracy of the different types of recommendation. This makes using the neural networks for the CAMS the classify the context of the user retrieved from the IoT.

Currently, the all the recommendation system work on only same type of recommendation. The proposed system recommending two type (song and movie) context. The context is retrieved virtually from the Internet of Thing which contain all the information of the user.

Further work may include improving recommendation by adding the score threshold based in the user's feedback. Moreover, the recommendation is given proactively to the user even without asking the user based on the mood of the user and the previous history of the user.

Most recommender system follow a request- response approach in which the recommendations are provided to the user upon his request. Recently a proactive recommender system - that pushes recommendations to the user when the current situation seems appropriate, without explicit user request. The fact that the future is for Internet of things, and the emergence of proactivity concept in which multi- type rather than one type of recommendations will be recommended proactively to the user in real time.

The proactive recommender system will be intelligent enough to recommend user the appropriate thing by analyze the context of the user without even asking from the user.

References

[1] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, Recommender systems - an introduction. Cambridge University Press, 2010.

[2] M. Balabanovic and Y. Shoham, “Content-based, collaborative recommendation,” Communications of the ACM, vol. 40, no. 3, pp. 66–72, 1997.

[3] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “Grouplens: An open architecture for collaborative filtering of netnews,” in CSCW. ACM, 1994, pp. 175–186.

[4] Z. Huang, D. Zeng, and H. Chen, “A comparative study of recommendation algorithms for e-commerce applications,” IEEE Intelligent systems, vol. 22, no. 5, pp. 68–78, 2007.

[5] G. Shani and A. Gunawardana, “Evaluating recommendation systems,” in Recommender systems handbook. Springer, 2011, pp. 257–297.

[6] M. Weiser, The computer for the 21st century, Sci. Am. (1991) 94–100.

[7] M. Presser, A. Gluhak, The Internet of Things: Connecting the Real World with the Digital World, EURESCOM mess@ge – The Magazine for Telecom Insiders, vol. 2, 2009

[8] Adomavicius G. and Tuzhilin A. (2005) ‘Toward the Next Generation of Recommender Systems: A Survey of the State–of–the–Art and Possible Extensions’, *IEEE Transactions on Knowledge and Data Engineering*, **17**(6), 734 – 749.

[9] Schafer, J.B., Konstan, J.A., and Riedl, J. (2001) ‘E–Commerce Recommendation Applications’, *Data Mining and Knowledge Discovery*, **5**(1/2), 115 – 152.

[10] Kazienko, P., Kołodziejski, P. (2005) ‘WindOwls – Adaptive Sytems for the Integration of Recommendation Methods in E – commerce’, *Springer Verlag*, 218 – 224.

[11] Balm, G.J. (1996) ‘Benchmarking and gap analysis: what is the next milestone?’, *Benchmarking for Quality Management & Technology*, **3**(4), 28 – 33.

[12] Terveen, L., Hill W., Amento B., McDonald D. and Creter J. (1997) ‘PHOAKS: A system for sharing recommendations’, *Communications of the ACM*, 40 (3) 59 – 62

[13] Burke, R. (2002) ‘Hybrid Recommender Systems: Survey and Experiments’, *User Modeling and User Adapted Interaction*, **12**(4), 331 – 370.

[14] Montaner, M., Lopez, B. and De la Rosa J.L. (2003) 'A Taxonomy of Recommender Agents on the Internet', *Artificial Intelligence Review*, Kluwer Academic Publisher.

[15] Schafer, J.B., Konstan, J.A., and Riedl, J. (2001) 'E–Commerce Recommendation Applications', *Data Mining and Knowledge Discovery*.

[16] Resnick, P. and Varian, H. (1997) 'Recommender systems', *Communications of the ACM*,.

[17] Kazienko P. and Kiewra M. (2004) 'Personalized Recommendation of Web Pages', Chapter 10 in Intelligent Technologies for Inconsistent Knowledge Processing, Nguyen T., (ed.), Advanced Knowledge International, Adelaide, South Australia.

[18] Pazzani, M. (1999) 'A Framework for Collaborative, Content–Based and Demographic Filtering', *Artificial Intelligence Review*,.

[19] Kruwlich, B. (1997) 'Lifestyle Finder: Intelligent User Profiling Using Large–Scale Demographic Data', *Artificial Intelligence Magazine*.

[20] Rich, E. (1979) 'User modeling via Stereotypes', *Cognitive Science*.

[21] Goldberg, D., Nichols, D., Oki, B.M. and Terry, D.B. (1992) 'Using Collaborative Filtering to Weave an Information Tapestry', *Communications of the ACM*.

[22] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J. (1994) 'GroupLens: An Open Architecture for Collaborative Filtering of Netnews', *Proceedings of the Conference on Computer Supported Cooperative Work*.

[23] Shardanand, U. and Maes, P. (1995) 'Social Information Filtering: Algorithms for Automating "Word of Mouth"', *Proc. Conf. Human Factors in Computing Systems*.

[24] Shardanand, U. and Maes, P. (1995) 'Social Information Filtering: Algorithms for Automating "Word of Mouth"', *Proc. Conf. Human Factors in Computing Systems*.

[25] Hill, W., Stead, L., Rosenstein, M. and Furnas, G. (1995) 'Recommending and Evaluating Choices in a Virtual Community of Use', *Proc. Conf. Human Factors in Computing Systems*.

[26] Lang, K. (1995) 'Newsweeder: Learning to filter news', *Proceeding of the 12th International Conference on Machine Learning*.

[27] Pazzani, M., Muramatsu, J., Billsus, D. (1996) 'Syskill & Webert: Identifying Interesting Web Sites', *Proceeding of the 13th National Conference on Artificial Intelligence*.

[28] Mladenic, D. (1996) ‘Personal WebWatcher: Implementation and Design’, *Technical Report IJS–DP–7472, Department of Intelligent Systems, Slovenia: J. Stefan Institute*.

[29] E. Selberg. *Towards Comprehensive Web Search*. PhD thesis, University of Washington, 1999.

[30] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative ¯ltering. ACM Press, http://doi.acm.org/10.1145/312624.312682, 1999.

[31] Internet usage statistics - The big picture of world Internet users and population stats. http://www.internetworldstats.com/stats.htm.

[32] Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, pages 79_86. Association for Computational Linguistics, 2002.

[33] Dave, S. Lawrence, and D.M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In Proceedings of the 12th international conference on World Wide Web, pages 519_528. ACM, 2003. ISBN 1581136803.

[34] C. Na, H. Sui, C. Khoo, S. Chan, and Y. Zhou. Effectiveness of simple linguistic processing in automatic sentiment classification of product reviews. AD-VANCES IN KNOWLEDGE ORGANIZATION, 9:49_54, 2004. ISSN 0938-5495.

[35] Pang and L. Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2):1_135, 2008. ISSN 1554-0669.

[36] D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classificationof reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pages 417_424. Association for Computational Linguistics, 2002.

[37] Godbole, M. Srinivasaiah, and S. Skiena. Large-scale sentiment analysis for news and blogs. In Proceedings of the International Conference on Weblogs and Social Media (ICWSM). Citeseer, 2007.

[38] Liu, X. Li, W.S. Lee, and P.S. Yu. Text classification by labeling words. In PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL

INTELLIGENCE, pages 425_430. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[39] Prabowo and M. Thelwall. Sentiment analysis: A combined approach. Journal of Informetrics, 3(2):143_157, 2009. ISSN 1751-1577.

[40] Yang, L. Si, and J. Callan. Knowledge transfer and opinion detection in the TREC2006 blog track. In Proceedings of TREC, volume 120. Citeseer, 2006.

[41] Pang and L. Lee. Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2):1_135, 2008. ISSN 1554-0669.

[42] Das and M. Chen. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA), 2001.

[43] Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford, 2009.

[44] Barbosa and J. Feng. Robust sentiment detection on Twitter from biased and noisy data. In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 36_44. Association for Computational Linguistics, 2010.

[45] Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of LREC 2010, 2010.

[46] A. Honkela, Nonlinear switching state-space models.

URL ttp://www.hiit.fi/u/ahonkela/dippa/dippa.html

[47] S. Haykin, Neural Networks and Learning machines, Pearson Education, 2009.